Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Sistema de calibración pre-impresión para imágenes DICOM

Estudante: Sergio Pardiñas Remeseiro
Dirección: Julián Alfonso Dorado De La Calle
Dirección: Daniel Sánchez Marzoa

A Coruña, febreiro de 2020.

Agradecimientos

Agradecimientos a M3M por proveer la infraestructura necesaria para la realización de este proyecto. Agradecimientos también a todo el personal por tratarme tan bien y ayudarme cuando lo he necesitado.

Resumen

Hoy en día las imágenes médicas se crean digitalmente en formato DICOM, sin embargo, muchos estudios se imprimen en papel para su distribución. Este proyecto se realizará para una empresa que, entre otras cosas, provee a los hospitales un sistema de impresión de imágenes DICOM. El principal problema es que muchos de los estudios, debido a la cantidad de detalles de las imágenes, no se visualizan correctamente en papel, lo que provoca que no se pueda interpretar el diagnóstico. El objetivo de este proyecto es estudiar el sistema actual e investigar el proceso de impresión para conseguir mantener la calidad y el detalle que permiten realizar un diagnóstico. Tras realizar dicha investigación e implementar en el sistema actual los cambios necesarios, se desarrollará una herramienta gráfica que permita al usuario calibrar una imagen antes de imprimirla, para así asegurar la mejor calidad posible tras la impresión.

Abstract

Today, medical images are digitally created in DICOM format. However, many studies are printed on paper for distribution. This project will be developed for a company that provides a DICOM Print Service for hospitals. Main problem is that, due to the amount of details in the images, many of the studies are not correctly displayed on paper, which means diagnosis cannot be interpreted by professionals. The main purpose of this project is to study the current system and investigate the printing process to preserve the quality and level of details that allow diagnosis. After concluding research and implementing the necessary changes in the current system, a graphical tool will be developed, allowing an user to calibrate an image before printing it, in order to ensure the best possible quality after printing it.

Palabras clave:

- DICOM
- PACS
- Imágenes médicas
- Impresión de imágenes
- Servicio de impresión DICOM
- Calibración de imágenes médicas
- Calibración DICOM

Keywords:

- DICOM
- PACS
- Medical images
- Image printing
- DICOM Print Server
- Medical images calibration
- DICOM Calibration

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Actualmente las imágenes médicas se producen y distribuyen en formato digital, siguiendo el estándar DICOM (Digital Imaging and Communication On Medicine). Cada archivo DICOM se compone de una serie de ”tags” o etiquetas acerca del estudio y del paciente. La etiqueta más relevante para este trabajo es ”Pixel Data”, la cual contiene los valores de los píxeles de la imagen médica. Cabe destacar que dichos valores no se corresponden con los formatos tradicionales de imagen, lo cual supone un problema al intentar convertir esa imagen médica representada en el Pixel Data en una imagen que pueda ser interpretada por una pantalla o impresora convencional. Para entenderlo con un ejemplo, podríamos decir que en una ecografía, cada uno de los píxeles del ”Pixel Data” representa la distancia a la que se encuentra el elemento en el que ha rebotado la onda emitida por la máquina. Tanto el rango de valores como la interpretación que se debe dar a los mismos depende de cada modalidad.

Para poder visualizar las imágenes, los radiólogos utilizan unas pantallas especiales específicamente creadas y preparadas para el diagnóstico, como por ejemplo las comercializadas por la marca Barco. Estas pantallas tienen una muy alta resolución y suelen tener unas herramientas propias para mostrar y mejorar automáticamente la imagen DICOM que se visualiza; sin embargo, hay ocasiones en las que la distribución se realiza en papel, lo cual implica una importante conversión de soporte que debe realizarse con la menor pérdida de calidad posible. Dicha distribución en papel es especialmente útil entre distintos hospitales, ya que un paciente puede necesitar hacer una prueba médica y ser diagnosticado en ese hospital, pero necesitar acudir a su médico prescriptor para recibir una prescripción médica. Para ello, la opción más cómoda y económica es imprimir las imágenes y entregárselas al paciente para que se las lleve a su médico.

Aunque la distribución en papel sea la opción más cómoda, acarrea una serie de problemas directamente relacionados con la impresión de la imagen. Si bien es cierto que, por ejemplo, en una radiografía de un hueso roto se puede apreciar claramente en la imagen impresa la rotura del hueso, en muchos estudios la información más importante de una imagen suele

residir en los detalles de la imagen, ciertas zonas pequeñas o que pueden tener tonalidades muy parecidas, por lo que si se imprimen las imágenes en una impresora normal o utilizando un método de impresión corriente, es imposible poder apreciar esos detalles. Es imperativo que durante el proceso de impresión, la imagen no pierda calidad en dichos detalles, ya que el médico prescriptor podría interpretar incorrectamente el diagnóstico, y al no poder apoyarse en la imagen impresa por no poder apreciar los detalles, cometer algún error que repercuta directamente en la salud del paciente.

El proyecto ha sido propuesto por una empresa y realizado mediante un convenio de prácticas. Dicha empresa provee a distintos hospitales un sistema que, entre otros servicios, permite imprimir las imágenes médicas de un mismo estudio en el formato que el hospital desee. El objetivo principal del proyecto es investigar la posibilidad de mejorar la impresión de las imágenes, para tratar de llegar a un punto donde la imagen impresa se parezca lo máximo posible a la que se muestra en pantalla. De existir dicha posibilidad, se pide también desarrollar un sistema que permita implementar la solución en el sistema ya existente.

Capítulo 2

Fundamentos

En este capítulo se explicarán los conceptos básicos necesarios para poder entender mejor el desarrollo del proyecto, así como una pequeña explicación de cada una de las herramientas utilizadas.

2.1 Conceptos Básicos

2.1.1 Estándar DICOM

DICOM[1] — Digital Imaging and Communications in Medicine — es el estándar internacional para imágenes médicas e información relacionada. Define el formato que deben seguir las imágenes para ser distribuidas con la información y calidad necesarias para uso clínico. Desde su primera aparición en 1993, este estándar ha revolucionado la radiología, permitiendo reemplazar las películas de rayos X por un modelo de trabajo totalmente digital.

Hoy en día, el estándar DICOM se usa prácticamente en cualquier ámbito médico, ya que es uno de los estándares para la distribución de imágenes médicas más extendidos mundialmente.

DICOM agrupa la información en grupos de datos. Esto quiere decir que un archivo de una imagen de rayos X, por ejemplo, contiene tanto la información del paciente como la información de la imagen dentro del mismo archivo, por lo que la imagen y la información no pueden ser separados por error.

Estos grupos de datos se llaman atributos o ”tags” (ver punto 2.1.1), que pueden ser el nombre del paciente, su Id, Id del estudio, etc. Además existe un atributo especial, denominado ”Pixel Data”, que contiene el valor de cada píxel de la imagen. Un objeto DICOM solo puede tener un ”Pixel Data”, lo cual se correspondería en muchas modalidades con una única imagen. Sin embargo, el atributo podría contener varios ”frames”, lo cual permite guardar una especie de película de ciertos estudios.

El ”Pixel Data” no tiene un rango de valores definido, ya que su rango de valores depende

de la modalidad del estudio y la máquina que haya producido el archivo DICOM. Para poder representar una imagen DICOM en una pantalla, se usa lo que llamamos ”Look Up Tables (LUT)”, las cuales nos permiten interpolar los valores del ”Pixel Data” a valores, típicamente, entre 0 y 255. Debido a que la información del ”Pixel Data” suele tener solo una dimensión, las imágenes se representarán típicamente en escala de grises.

DICOM Tags

Como ya se ha explicado anteriormente, los archivos DICOM están compuestos por tags. El estándar recoge una enorme cantidad de tags, pero solo una pequeña parte son obligatorios. Toda la información de los tags DICOM se recoge en el apartado 3.6 del estándar[2].

Cada tag se identifica por un código de ocho dígitos en hexadecimal de la forma (0000,0000), donde los cuatro primeros dígitos se corresponden con el número del grupo, que llamaremos ”grupo DICOM”, y los cuatro últimos definen el elemento dentro de dicho grupo. Sin embargo, para comprenderlo mejor nos referiremos a ellos por su ”Keyword”. Por ejemplo, en el caso del nombre del paciente, el tag es (0010,0010) y el keyword es Patient’s Name.

Para este proyecto, podemos agrupar los tags más importantes en cuatro grupos:

- **Paciente:** Este grupo de tags se agrupan en un único grupo DICOM identificado por el código 0010, que recoge toda la información del paciente, como por ejemplo su nombre *(0010,0010)Patient’s Name*, su Id *(0010,0020)Patient’s Id*, su sexo *(0010,0040)Patient’s Sex*, etc.

- **Estudio:** Los datos acerca del estudio se suelen recoger en el grupo DICOM 0008, como pueden ser la fecha del estudio *(0008,0020)Study Date*, la hora en la que comenzó *(0008,0030)Study Time*, el médico que la realizó *(0008,0090)Referring Physician’s Name* o la modalidad[3] *(0008,0060)Modality*, que es el tipo de equipamiento que se utiliza para realizar el estudio, o lo que es lo mismo, el tipo de estudio realizado.

- **Serie:** Las series son agrupaciones de imágenes dentro de un mismo estudio. Los datos sobre las series se suelen agrupar también en el grupo DICOM 0008, como pueden ser por ejemplo la fecha *(0008,0021)Series Date* o la descripción de las series *(0008,103E)Series Description*.

- **Imagen:** Para este trabajo, éste es el grupo más importante, ya que recoge todos los atributos de la imagen. Dichos atributos se recogen en el grupo DICOM 0028, y entre ellos se encuentran el alto y ancho de la imagen *(0028,0010)Rows* y *(0028,0011)Columns*, los valores extremos del Pixel Data *(0028,0106)Smallest Image Pixel Value* y *(0028,0107)Largest Image Pixel Value*, el Windowing (ver punto 2.1.1) *(0028,1051)Window Width* y *(0028,1050)Window Center*

El Pixel Data se recoge en un grupo DICOM especial (7FE0), con el tag *(7FE0,0010)Pixel Data.*

Windowing

El Windowing se compone de dos atributos o ”tags” recogidos en el estándar DICOM, llamados Window Width y Window Center. Dichos atributos nos sirven para delimitar los valores del ”Píxel Data” que queremos que se interpolen en las ”Look Up Tables (LUT)” (ver apartado 2.1.1).

Por ejemplo, supongamos que tenemos un archivo DICOM con un ”Pixel Data” cuyo rango de valores va desde 0 a 4000, siendo 0 el valor más oscuro y 4000 el valor más claro. Lo habitual es que parte de la imagen sea un ”fondo” negro, el cual puede estar formado por valores de 0 a 100. El Windowing nos sirve para indicar que en la ”Look Up Table” queremos que todos los valores del 0 al 100 se interpolen como un valor 0 (negro), permitiendo así un mayor rango dinámico en el resto de la imagen, lo cual nos ayuda a ver mejor los detalles.

Look Up Tables

Valores del pixel data	Valores de gris
0	0
1	0
2	0
3	1
4	1
…	…
1000	87
1001	88
1002	88
1003	88
…	…
3998	255
3999	255
4000	255

Tabla 2.1: Ejemplo de tabla LUT

La LUT es una tabla que determina la interpolación que se debe realizar entre el tag DICOM ”Pixel Data” y los valores de gris de la imagen final, ya que dichos valores deben ser representados en un formato comprensible para cualquier sistema. Es decir, para cada valor del pixel data, determina su homólogo en valores de 0 a 255. La tabla 2.1 es parte de una tabla LUT, en la cual, como se puede observar, los valores del pixel data tienen un rango de 0 a

4000. Como es de suponer, una variación en el windowing implica también una variación en la tabla LUT, ya que solamente los valores dentro del rango del windowing se interpolan a valores de gris. Los valores fuera del rango del windowing se interpolan a negro (0) o blanco (255).

Modalidad

La modalidad representa el tipo de estudio, y viene definida por la técnica usada para realizar el estudio. Algunos ejemplos de modalidad son: TAC, radiografía, resonancia o ecografía.

2.1.2 Book

En el ámbito médico, un book es una especie de libro donde se imprimen las imágenes de un cierto estudio. Generalmente, además de las imágenes, se añade una hoja extra para incluir el diagnóstico.

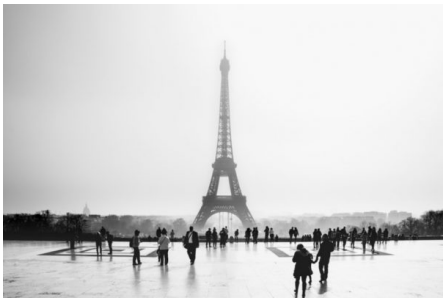
Se suele utilizar cuando a un paciente se le realiza un estudio en un hospital y luego necesita presentar dicho estudio en otro, ya que es un método muy barato y apenas requiere esfuerzo por parte de los médicos.

2.1.3 Manipulación de imágenes

Histograma

El histograma es un gráfico que muestra la cantidad de píxeles de un mismo tono de gris que hay en una imagen. Por ejemplo, en la figura 2.1a se muestra el histograma de la figura 2.1b. Como podemos observar, prácticamente todos los píxeles de la imagen se encuentran en los extremos del histograma, eso significa que la mayor parte de valores de gris son extremos, es decir, o negro o blanco.

(a) Histograma



(b) Imagen

Figura 2.1: Ejemplo del histograma de una imagen en escala de grises

Brillo

En manipulación de imágenes, el brillo se utiliza para aumentar o disminuir la luminosidad de la imagen. A efectos prácticos, aumentar el brillo significa aumentar el valor de los píxeles de la imagen, aclarando así los valores más oscuros y blanqueando los claros.

Contraste

En manipulación de imágenes, el contraste ajusta la diferencia entre los colores más claros y los más oscuros. A efectos prácticos, aumentar el contraste significa oscurecer los píxeles oscuros y aclarar los claros.

Modelo de color

Un modelo de color es un modelo matemático que permite la representación de los colores de una forma numérica. Esta representación suele formarse con tres o cuatro elementos cromáticos, como por ejemplo RGB o CMYK.

Los modelos de color principales son:

- **RGB (Red Green Blue):** basado en la síntesis aditiva, es posible representar un color mediante la mezcla por adición de los tres colores primarios. Es el modelo que suelen usar los sistemas que emiten luz propia, como los monitores, las televisiones o los proyectores.
- **CMYK (Cian Magenta Yellow Key:** basado en la síntesis sustractiva, se basa en la mezcla de pigmentos para crear nuevos, ”eliminando” colores al blanco. ”Key” es para el color negro, y éste se usa para evitar la mezcla de los otros tres pigmentos.
- **HSL (Hue Saturation Lightness):** es una deformación no lineal del modelo de color RGB que se usa para el tratamiento del color.
- **HSV (Hue Saturation Value):** Al igual que HSL, es una deformación no lineal del modelo de color RGB que se usa para el tratamiento del color.

Espacio de color

Un espacio de color es una organización específica de los colores de una imagen. Cada espacio de color se basa en un modelo de color para reproducir los colores. Por ejemplo, Adobe RGB y sRGB son dos espacios de color diferentes basados en el modelo RGB. Debido a las propiedades de cada dispositivo, su espacio de color puede ser diferente y por tanto su representación de los colores.

Perfil de color

Los perfiles de color son un conjunto de datos que caracteriza a un dispositivo de entrada o salida de color. Cada perfil describe los atributos de color de un dispositivo en particular mediante tablas, en las cuales se aplica interpolación. Los perfiles nos permiten cambiar entre espacios y modelos de color mediante dichas tablas.

Cada dispositivo que captura o muestra color puede tener su propio perfil. Algunos fabricantes suministran perfiles para sus productos, y hay algunos que permiten a los usuarios generar sus propios perfiles.

2.1.4 Negro Puro y Negro Compuesto

Negro puro y negro compuesto son dos opciones distintas de generación de negro con impresoras. Negro compuesto significa mezclar los pigmentos CMY para generar negros y grises, sin utilizar pigmento negro. Negro puro significa lo opuesto, generar los negros y grises con pigmento negro.

Mientras que el con el negro compuesto se pueden obtener negros más puros e imágenes con mayor profundidad, el negro puro nos permite un ahorro de tinta considerable.

2.2 Herramientas Usadas

2.2.1 Lenguajes de Programación y de Marcado

Java

Java[4, 5, 6] es un lenguaje de programación de alto nivel orientado a objetos, cuya sintaxis deriva en gran medida de C y C++. Las aplicaciones de Java cuentan con un entorno de ejecución propio o máquina virtual (JVM), lo cual permite ejecutar un programa creado en Java en cualquier sistema que soporte dicho entorno de ejecución sin importar su arquitectura.

Ha sido comercializada por primera vez en 1995 por Sun Microsystems, pero pertenece a Oracle desde 2010.

Python

Python[7, 8] es un lenguaje de programación interpretado, de alto nivel y orientado a objetos. Fue creado a principios de 1990 por Guido van Rossum, y desde siempre ha sido de código abierto.

Python fue creado con el propósito de ser un lenguaje fácilmente legible, por eso algunas de sus principales características son el uso de ”indentación” o sangrado para delimitar blo-

ques en lugar de llaves, la no necesidad de poner puntos y coma al final de una sentencia o el tipado dinámico.

JavaScript

JavaScript[9] es un lenguaje interpretado, de alto nivel, orientado a objetos y con tipado dinámico.

Junto con HTML y CSS, conforman las tecnologías en las que se basan prácticamente todas las páginas web hoy en día, y aunque inicialmente solo se usaba en navegadores, muchos entornos actuales permiten el uso de JavaScript, como por ejemplo Node.js o Adobe Acrobat.

En 1997, ECMA International publicó ECMAScript[10], creado para estandarizar JavaScript, así como otros lenguajes.

HTML

HTML[11, 12] (HyperText Markup Language) es un lenguaje de marcado para la elaboración de páginas web. HTML se utiliza para crear la estructura de una página web. Es un estándar marcado por el World Wide Web Consortium (W3C)[13] y que todos los navegadores han adoptado.

Al ser un lenguaje de marcado, su elemento principal son las etiquetas. Gracias a dichas etiquetas se puede definir la estructura de la página o incluso incrustar ciertos elementos como imágenes o scripts.

CSS

CSS[12, 14] (Cascading Style Sheets) es un lenguaje de diseño gráfico utilizado para definir el diseño de un documento con una estructura definida por un lenguaje de marcado, como puede ser HTML.

Las especificaciones de CSS son definidas por el World Wide Web Consortium (W3C) y todos los navegadores actuales son capaces de interpretarlo.

2.2.2 Frameworks de desarrollo

Spring

Spring[15, 16] es un framework de desarrollo para Java, principalmente utilizado para la creación de aplicaciones web.

La primera versión fue escrita por Rod Johnson, en octubre del 2002. Pero no fue hasta marzo de 2004 cuando se lanzó la versión 1.0.

Una de las principales características de Spring es su ”programación orientada a aspectos” (AOP)[17], que es un paradigma de programación que complementa a la programación

orientada a objetos y que permite su modularización. Por tanto, podemos decir que Spring está compuesto por diferentes módulos.

Jest

Jest[18] es un framework mantenido por Facebook para la generación y ejecución de pruebas en JavaScript.

JUnit

JUnit[19] es un framework para la generación y ejecución de pruebas en Java.

2.2.3 Librerías, extensiones y servicios

Tomcat

Tomcat[20] es una implementación de código abierto que permite ejecutar en java un servidor web HTTP. Fue lanzado en 1999 y es mantenido por una comunidad abierta de desarrolladores de "Apache Software Foundation"[21].

2.2.4 Apache HTTP Server

Apache HTTP Server[22] es un servidor HTTP de código abierto multiplataforma desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation[21]

Dcm4Che

Dcm4che[23] es una colección de utilidades de código abierto desarrolladas para Java con el objetivo de producir aplicaciones relacionadas con el mundo médico.

d3js

D3js[24] es una ibrería para JavaScript que permite generar tablas o gráficos a partir de grandes cantidades de datos de una forma sencilla.

JQuery

Hoy en día, y debido a la cantidad de problemas inherentes a JavaScript, existen multitud de librerías que permiten al desarrollador crear una aplicación de una forma más sencilla. Entre ellas, se encuentra la librería utilizada para este proyecto, JQuery[25, 26].

JQuery es una librería que facilita la manipulación de los elementos HTML de la página, así como los eventos, animaciones, etc. El propósito principal para la utilización de esta librería es AJAX, un método que nos permite hacer peticiones GET y POST de una forma sencilla.

2.2.5 Entornos de trabajo o IDEs

Spyder

Spyder[27, 28] es un entorno de trabajo multi-plataforma escrito en phyton y para programar en python. Además, Spyder integra un gran número de paquetes para trabajar en python, como NumPy, SciPy, Matplotlib, etc.

Entre sus principales características se encuentra un editor de texto con resaltado de sintaxis, introspección de tipos, completado de código, consola python para la ejecución interactiva de código, un historial de comandos y hasta un ”debugger”.

Visual Studio Code

Visual Studio Code[29] es un editor de código fuente multiplataforma desarrollado por Microsoft, anunciado en abril de 2015 y lanzado el 14 de abril de 2016[30].

Por defecto, Visual Studio Code posee resaltado de sintaxis, autocompletado con IntelliSense [31] (popups de autocompletado mientras se escribe, mostrar los parámetros necesarios para las funciones o pistas relacionadas con errores de sintaxis) incluso de módulos importados, debug y ejecución de aplicaciones directamente en el editor, consola de comandos incorporada para mostrar la ejecución de las aplicaciones o incluso ejecutar comandos, etc.

A pesar de no incorporar por defecto soporte para todos los lenguajes de programación, una de las principales ventajas de este editor es que posee una pestaña dedicada exclusivamente a las extensiones. Dichas extensiones son muy fáciles de buscar e instalar, y proporcionan desde soporte para lenguajes que no están en la instalación por defecto, hasta compatibilidad con herramientas o frameworks externos como Docker, Maven o Spring.

2.2.6 Herramientas para la creación de diagramas y prototipos

StarUML

StarUML[32] es un software de modelado compatible con el estándar UML que permite un desarrollo rápido de diagramas para el diseño de un producto software.

GIMP

GIMP[33] es un programa diseñado para la edición de imágenes gratuito y de código abierto.

Vectr

Vectr[34] es una herramienta web gratuita para crear imágenes vectoriales de una forma fácil e intuitiva. Para desarrollo software es especialmente útil para el desarrollo de prototipos.

2.2.7 Herramientas para la edición y composición de textos

LaTeX

LaTeX[35] es un sistema de composición de textos de código abierto, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. LaTeX es el estándar para la comunicación y publicación de documentos científicos

Overleaf

Overleaf[36] es un editor online gratuito para la creación y modificación de documentos LaTeX.

2.2.8 LibreOffice

LibreOffice[37] es una suite gratuita de herramientas de oficina, entre las que se incluyen herramientas para la creación de documentos de texto, hojas de cálculo o hasta gráficos vectoriales.

2.2.9 Herramientas para la visualizacion y transmisión de archivos DICOM

Weasis

Weasis[38] es un visualizador de imágenes DICOM de software libre que permite ver las imágenes de los archivos en alta resolución con un alto rendimiento.

Conquest

Conquest[39] es un software que implementa un servidor para el almacenamiento y transmisión de archivos DICOM.

2.2.10 Impresoras

Las impresoras utilizadas han sido la Xerox Color 550[40], la Xerox C8530[41] y la Xerox Phaser 7760[42]. Todas ellas son impresoras de la marca Xerox[43], mayor proveedor mundial de impresoras láser. Son de las mejores impresoras actualmente en el mercado, con una enorme calidad de impresión y resolución (hasta 2400x2400 puntos por pulgada).

Las tres impresoras son modulares y cuentan con módulos finalizadores, lo cual permite que la impresora automáticamente doble hojas y las grape para la creación de Books.

Para hacernos una idea de las dimensiones de las impresoras, con finalizadores miden cerca de dos metros de ancho, casi metro y medio de alto y un metro de profundidad. En las figuras 2.2 se pueden apreciar las dimensiones de las impresoras.

(a) Xerox Color 550

(b) Xerox C8035

(c) Xerox Phaser 7760

Figura 2.2: Impresoras disponibles

Capítulo 3

Estado de la Cuestión

El sistema a mejorar se denomina DPS (DICOM Printer Service). Dicho sistema se encarga de gestionar las peticiones de impresión de archivos DICOM. Cuando recibe una petición, el DPS la procesa y antes de imprimirlo guarda una copia del trabajo en el historial.

Debido a que el problema para imprimir correctamente las imágenes ya existía antes, el DPS posee una herramienta de calibración. Dicha herramienta permite al usuario modificar el brillo, el contraste, el windowing y la LUT de la imagen DICOM para mejorar su impresión y que se parezca más a las imágenes en pantalla. En el punto 3.2 se explicará el proceso de calibración y su importancia más en profundidad.

En la figura 3.1 se muestra el proceso actual desde la generación de un archivo DICOM hasta su impresión en papel: La modalidad genera el archivo DICOM, éste se envía al DPS para su impresión, el DPS extrae la imagen del archivo junto con los datos del paciente, genera un pdf y la impresora lo imprime directamente en formato de Book. Como se puede observar, la herramienta de calibración interviene antes de la extracción de información del archivo DICOM.

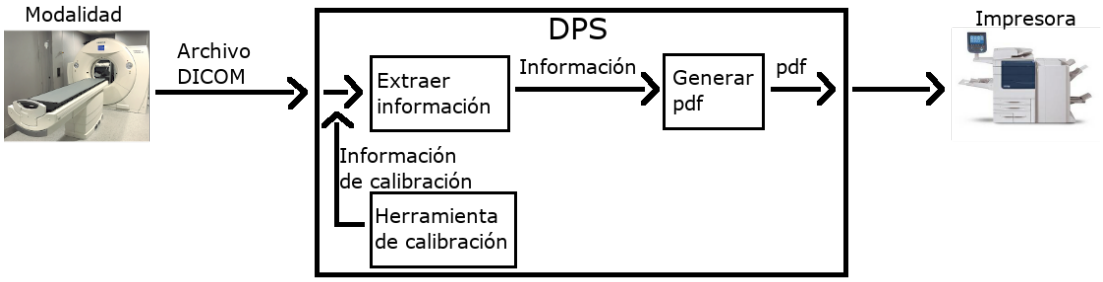


Figura 3.1: Proceso de impresión a través del DPS

La información que se extrae del archivo DICOM incluye también información sobre el paciente y el estudio, pero son irrelevantes e independientes a la impresión, por lo que para el estudio y posterior desarrollo de este sistema supondremos que solo extrae la información

de la imagen médica.

A continuación se explicará más a fondo el funcionamiento del DPS descrito en la figura 3.1, así como su herramienta de calibración. Al final del capítulo se explicarán los resultados generales del sistema y consideraciones para el desarrollo del proyecto.

3.1 Funcionamiento del DPS

El proceso del sistema para imprimir la imagen de un archivo DICOM es el siguiente:

- Aplicar modificaciones de windowing y LUT sobre el archivo DICOM de ser preciso (ver punto 3.2).
- Extracción de la imagen del archivo DICOM y guardado en archivo png.
- Si el estudio está compuesto por más de una imagen, se crea un ”grid” en formato png con las imágenes del estudio y se guarda.
- En una plantilla pdf, se inserta el grid.
- Se juntan todos los pdf, en caso de haber más de uno y se manda imprimir.

En la figura 3.2 se puede observar gráficamente el proceso de impresión de la imagen DICOM.

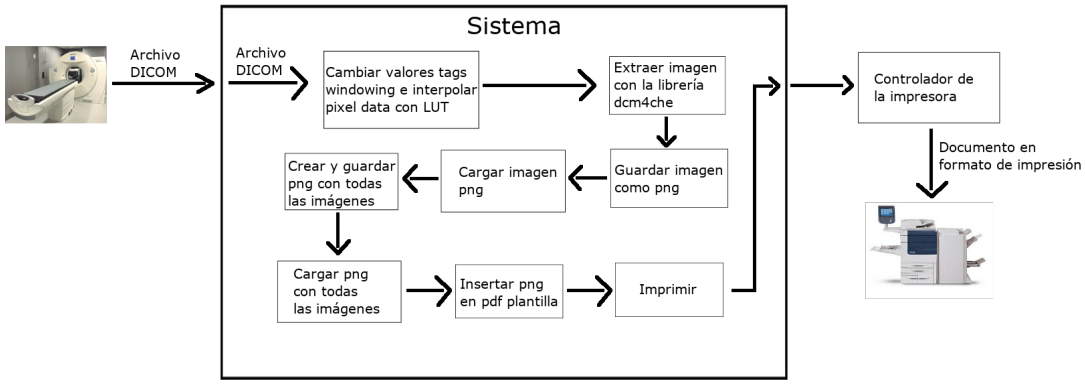


Figura 3.2: Proceso previo

Aplicar Calibración

Para aplicar la calibración, el sistema primero cambia los tags pertenecientes al windowing y establece los indicados por el usuario. Después, cambia uno a uno los píxeles del tag Pixel Data para realizar la interpolación establecida por la función LUT. Cabe destacar que, en esta

versión de la aplicación, la función LUT no se utiliza para interpolar los valores del Pixel Data a valores entre 0 y 255, sino que los interpola a otros valores dentro del mismo rango, por lo que no se genera una imagen como tal.

El brillo y el contraste general se aplicarán tras extraer la imagen. Ver punto 3.1

Extraer Imagen

Tras aplicar los parámetros de calibración Windowing y LUT, la imagen se extrae utilizando la librería dcm4che.

Se extrae como un objeto de clase ”Raster” y más tarde se convierte en ”BufferedImage”, con un espacio de color RGB. En este paso la imagen se guarda como archivo png y se le aplican los parámetros de brillo y contraste especificados en la calibración.

Crear Grid

LLamamos grid a un conjunto de imágenes de un mismo estudio situadas en forma de rejilla. Aunque la disposición de las imágenes depende de las preferencias del cliente, suelen disponerse en un 2x3 (dos columnas y tres filas).

Una vez todas las imágenes de un estudio se han extraído y guardado como archivos png, se crea un grid también en formato png con dichas imágenes. Para ello, se crea un archivo png vacío con unas dimensiones de una hoja A4, y cada una de las imágenes del estudio se escala al tamaño deseado y se sitúa en la posición deseada del grid. Después, el grid se guarda como un archivo png.

Insertar Grid

Tras crear y guardar todos los grids necesarios para imprimir el Book del estudio, se procede a insertar dichos archivos en un archivo pdf. Para ello, se usa un pdf ”plantilla”, el cual no es más que un formulario pdf cuyo único elemento es una imagen que ocupa el espacio que deberá ocupar el grid. El formulario pdf nos permite insertar elementos de una forma sencilla.

El sistema rellena el formulario insertando el png del grid en el pdf plantilla y lo guarda para imprimir.

Imprimir

El sistema carga con la librería PdfBox el pdf previamente guardado y lo imprime. Permite dos modos de impresión: con adobe, ejecutando en el bash un comando para imprimir el archivo con adobe acrobat reader, o con la clase PrinterJob de Java, la cual permite crear un objeto de la clase PrinterJob a partir de un archivo pdf e imprimirlo estableciendo ciertos parámetros de impresión básicos.

Es importante destacar aquí que el Book se imprime utilizando lo que se llama ”negro compuesto”, que es simplemente que la impresora genera el negro utilizando cian, magenta y amarillo. Esto es importante destacarlo porque es un requisito para el sistema y la causa de uno de los principales problemas: La aparición de manchas de color en las imágenes médicas.

3.2 Herramienta de Calibración

En algunos estudios, como por ejemplo una radiografía de un hueso roto, se puede apreciar claramente en la imagen impresa la rotura del hueso. Sin embargo, en muchos estudios la información más importante de una imagen suele residir en zonas pequeñas o con tonalidades muy parecidas, como es el caso de, por ejemplo, las radiografías de tórax (ver figura 3.3).

Figura 3.3: Imagen de tórax

En la figura 3.3 se muestra una radiografía de tórax sin calibrar. Las zonas de interés de estas imágenes se encuentran tanto en los huesos (costillas o columna) como en los pulmones y el corazón. Como se puede observar, las tonalidades dentro de la zona de los pulmones son muy parecidas, lo cual imposibilita diferenciar todos los detalles de los pulmones. Además, debido a lo ”quemada” que se muestra la imagen en las zonas claras, es también imposible siquiera diferenciar gran parte de los huesos.

Simplemente aplicando un windowing que solamente permita ver los valores más claros de la imagen (ver figura 3.4), podemos ver que se aprecian mucho mejor los huesos y los detalles en la zona del diafragma. En esta imagen nos ayudamos del histograma para poder ver qué valores del pixel data estamos cogiendo

Calibrar la imagen antes de imprimirla es fundamental para asegurar la correcta impresión de la imagen sobre el papel. Los monitores que se utilizan para el diagnóstico tienen herramientas para autocalibrarse y mostrar la mejor calidad de imagen en cualquier lugar, y

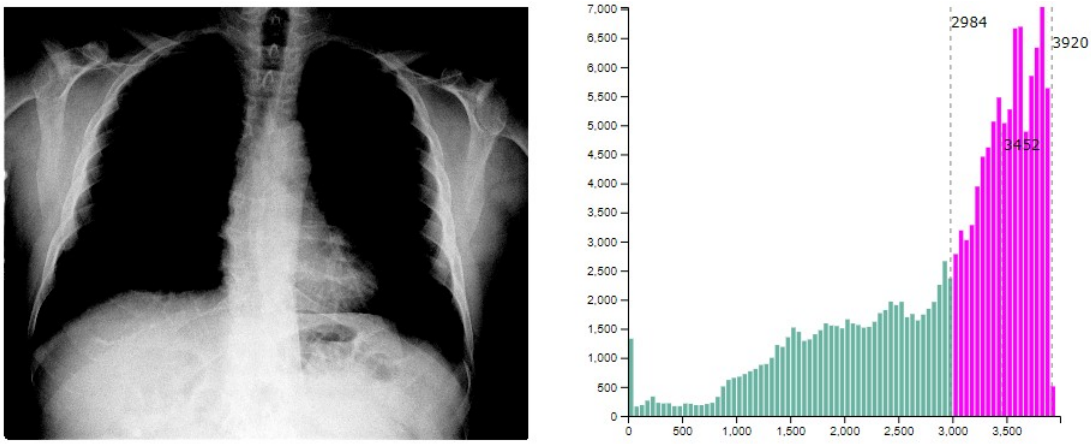


Figura 3.4: Imagen de tórax

además, las herramientas que utilizan los expertos les permiten modificar el windowing de la imagen para poder ver todo el rango de valores del pixel data; sin embargo, el papel, como es obvio, no dispone de dichas herramientas, por lo que debemos realizar una calibración previa a la impresión que nos permita ver todo lo que necesitamos.

En las figuras 3.5 se muestran ejemplos de archivos DICOM abiertos con el programa Weasis (ver punto 2.2.9). La imagen de la izquierda (figura 3.5a) tiene el windowing por defecto (window width 4096 y window center 2048), que en este caso recoge todos los valores de gris de la imagen. Simplemente arrastrando el ratón sobre la imagen, nos permite cambiar el windowing de una forma sencilla. En las figuras 3.5b y 3.5c se muestran cambios realizados en el windowing.

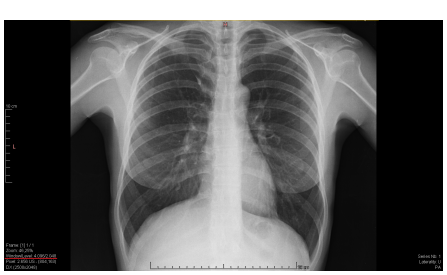
Para poder encontrar los mejores valores de windowing, LUT, brillo y contraste se creó una herramienta de calibración manual, donde el usuario puede ir modificando los valores y previsualizando el resultado en una imagen en el monitor. La herramienta es parte del DPS y está implementada en el navegador.

3.2.1 Proceso de calibración

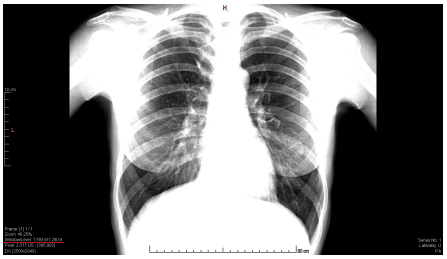
Cuando el usuario del DPS imprime una imagen médica y no puede distinguir las zonas importantes, comienza un proceso de calibración. En dicho proceso, el usuario indica manualmente al sistema los valores de windowing, LUT, brillo y contraste que desea aplicar sobre dicha imagen mediante una interfaz en el navegador (ver punto 3.2.2).

Mientras que el proceso de extracción de la imagen, generación del pdf e impresión se ejecutan en un servidor, la herramienta se ejecuta en un navegador, por lo que debe existir una comunicación constante entre ambos.

Para comenzar, el usuario escoge un estudio que desea calibrar, entonces, la herramienta



(a) Windowing por defecto (4096/2048)



(b) Disminuyendo window center y window width (1593/1280)

(c) Aumentando window center y disminuyendo window width (2851/2315)

Figura 3.5: Imágenes abiertas y modificadas con Weasis

pide al servidor una imagen representativa de dicho estudio y se la muestra al usuario y es entonces cuando el usuario comienza la calibración manual de la imagen. Para cada modificación que efectúa el usuario, ésta se envía al servidor para que aplique la modificación sobre el archivo DICOM y extraiga de nuevo la imagen para enviársela a la herramienta de calibración. En la figura 3.6 se puede apreciar el proceso cíclico.

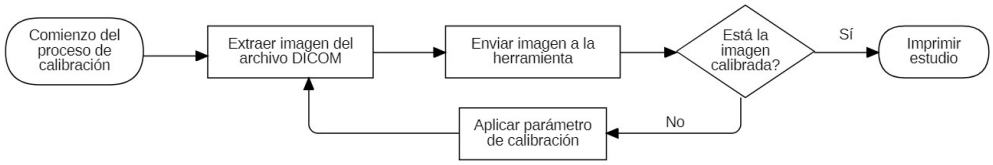


Figura 3.6: Diagrama de flujo de calibración

3.2.2 Interfaz de la herramienta de calibración

En la figura 3.7 se muestra la interfaz del sistema antes de la realización de este proyecto.

Como se puede observar, en la parte derecha se puede ver la imagen del estudio que sirve como referencia. En el centro, se pueden modificar parámetros como el windowing, el brillo y el contraste. Por último, a la izquierda se puede ver una función de interpolación, que no es más que una representación de la LUT. En los puntos 5.1.1 y 5.2.1 profundizaré más en ello.

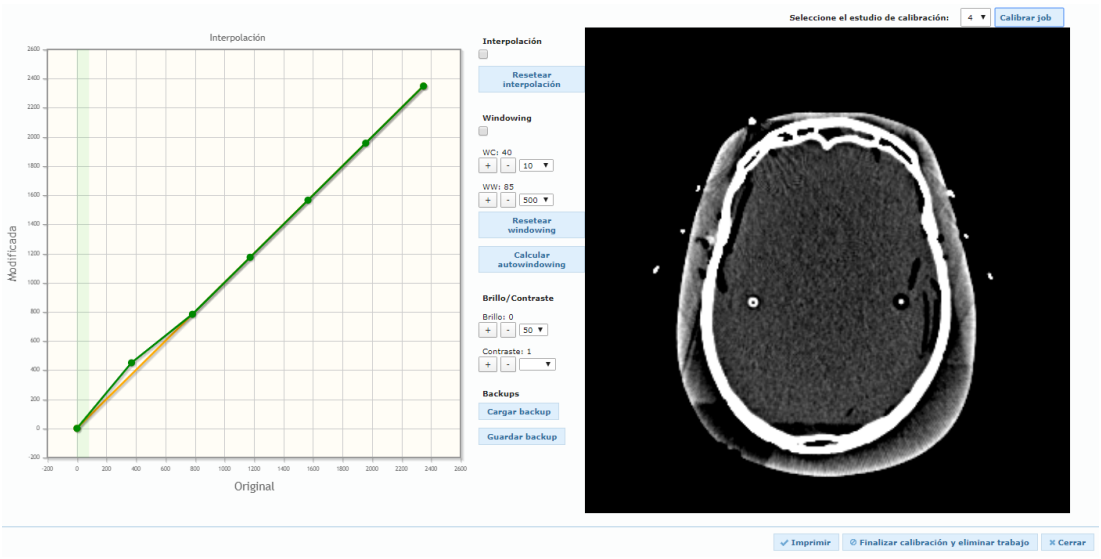


Figura 3.7: Interfaz gráfica de la herramienta de calibración

3.3 Resultados

Tras imprimir las imágenes, en algunos estudios puede ser difícil identificar los elementos que se encuentran en ellas, lo cual dificulta interpretar el diagnóstico del paciente. Además, se pueden apreciar tonalidades de color en las imágenes, a pesar de que éstas son en escala de grises.

Aunque las imágenes se muestran mejor tras calibrarlas, en algunos casos no es suficiente para identificar los detalles. Además, la herramienta de calibración tiene defectos:

- Por una parte, todos los cambios se realizan en el servidor, por lo que cada parámetro que se cambia ha de ser enviado y esperar a que el servidor mande de vuelta la imagen. Teniendo en cuenta que hay que modificar varios parámetros y buscar su valor ideal, hace que la tarea de calibración sea muy tediosa.

- Por otro lado, la única referencia para saber si los cambios realizados son correctos es la imagen que se muestra a la derecha. Es muy difícil saber a qué parte del pixel data están afectando los cambios si la única referencia está en otro rango de valores. Además, aunque en la imagen puedas intuir los cambios que has realizado, es muy difícil prever lo que ese cambio va a provocar en la imagen antes de efectuarlo.

- Además, la modificación de la función de interpolación ofrece demasiados grados de libertad, ya que cada punto de los que se pueden apreciar en la figura 3.7 se puede mover hacia cualquier dirección. Todo esto, unido a que es difícil de comprender su impacto sobre la imagen, hacen que sea muy difícil aprender a usarla.

3.4 Consideraciones

Para realizar una mejora al sistema, la principal consideración es investigar a fondo el proceso que lleva a cabo el sistema para encontrar el motivo por el que puede existir tanta diferencia entre la imagen visualizada en el monitor y la impresa en papel. Además, hay que considerar la búsqueda de nuevas herramientas que permitan mejorar o cambiar dicho proceso para obtener mejores resultados.

Además, de seguir siendo necesaria la herramienta de calibración, es imperativo modificarla o crear una nueva para que pueda ser del todo útil.

Capítulo 4

Metodología

PARA el desarrollo de este proyecto se ha utilizado un modelo incremental con dos iteraciones.

El modelo incremental está basado en el modelo tradicional en cascada pero su característica principal es la división en iteraciones. Dichas iteraciones, o incrementos, complementan a la iteración anterior suponiendo un avance con respecto a ella. Cada incremento comienza con el análisis y finaliza con un sistema completo.

Debido a que se basa en el modelo en cascada, para cada iteración se siguen las siguientes fases: análisis o especificación de requisitos, diseño, codificación y pruebas.

Al final del capítulo se describirán los patrones de diseño utilizados. Ver punto 4.5

4.1 Análisis / Especificación de requisitos

En el análisis se explicará todo el proceso seguido para comprender las necesidades del sistema. Debido a que el objetivo principal de este proyecto es investigar la mejor forma de imprimir imágenes DICOM, este paso es de vital importancia en la primera iteración.

En la segunda iteración, he hecho uso del modelo de prototipos para extraer sus requisitos y plasmarlos en casos de uso. Dicho modelo se basa en la creación de prototipos que deben ser construidos rápidamente; el cliente, o en este caso los responsables de la empresa, comprueban el prototipo y dan el visto bueno o critican lo que consideran ser cambiado para poder generar otro prototipo aplicando esos cambios. Este modelo es idóneo para esta segunda iteración, ya que su propósito principal es ser amigable con el usuario.

La notación de los casos de uso es UML[44]

La definición de los casos de uso se hará siguiendo el formato de la tabla de ejemplo 4.1. Dicha tabla es una simplificación de la que se explica en el libro de Alistair Cockburn[45], eliminando las celdas que no son necesarias para este trabajo.

CU-XX	
Nombre	Nombre del caso de uso
Actor	Actor que realiza el caso de uso
Objetivo	Objetivo principal del caso de uso
Precondiciones	Condiciones que han de existir para que el caso de uso se pueda llevar a cabo
Pasos	Pasos que se deberían seguir para llevar a cabo el caso de uso
Postcondiciones (opcional)	Cualquier suceso o acción que realice el sistema que pueda no ser intuitiva y por tanto haya que destacar
Excepciones	Refleja las acciones del sistema en caso de no poder realizar algún paso

Tabla 4.1: Tabla de Caso de Uso de ejemplo

4.2 Diseño

En el apartado de diseño se explicará mediante diagramas la estructura de cada una de las iteraciones del sistema, que servirá de guía para el siguiente paso de codificación. Los diagramas han sido realizados con la herramienta StarUML (ver punto 2.2.6).El apartado de diseño estará dividido en: diagrama de Clases, diagrama de Flujo de Datos, especificación de Procesos y diagrama de Secuencia.

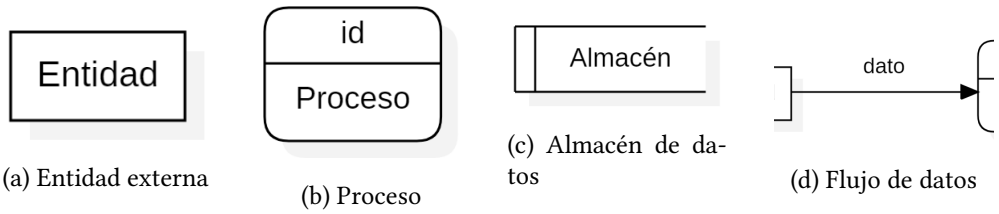
4.2.1 Diagramas de clases

Realizado en ambas iteraciones, describe la estructura del sistema mostrando las clases, sus atributos, métodos y relaciones directas. En cada apartado haré una breve descripción del propósito y utilidad de las clases más importantes. La notación utilizada es UML[46]. Ver puntos 5.1.2 y 5.2.2.

4.2.2 Diagrama de Flujo de Datos

Los diagramas de flujo de datos nos permiten entender gráficamente el flujo de procesado de los datos, dividiéndolo en procesos y estableciendo una comunicación entre ellos (flujo de datos). Los diagramas se dividen en niveles, y en cada nivel se amplía un proceso del nivel anterior (si necesita ser ampliado). Para el desarrollo de este sistema, solo he necesitado crear el diagrama de flujo de datos para la segunda iteración, debido a que la estructura de la primera iteración es simple y un diagrama de secuencia es suficiente.

La notación utilizada para el diagrama de flujo de datos es la disponible en StarUML, y como no es una notación estándar, creo conveniente especificar qué representa cada figura (figuras 4.1). Ver punto 5.2.2.



(a) Entidad externa (b) Proceso (c) Almacén de datos (d) Flujo de datos

Figura 4.1: Figuras de notación StarUML para un DFD

4.2.3 Especificación de Procesos

La especificación de procesos complementa al diagrama de flujo de datos, definiendo cada uno de los procesos que en él aparecen. Cabe destacar que por conveniencia solo se definen los del último nivel.

Para definir cada proceso, así como sus entradas y salidas he hecho uso de tablas como la del ejemplo 4.2. Ver punto 5.2.2.

Proceso X.X:	
Entrada	Entrada o entradas del proceso
Salida	Salida del proceso
Descripción	Definición abstracta del proceso, sin entrar en detalles

Tabla 4.2: Ejemplo de especificación de proceso

4.2.4 Diagrama de Secuencia

Los diagramas de secuencia nos ayudan a entender mejor la comunicación entre las clases del sistema mediante el desarrollo de cada una de las posibles interacciones. La notación utilizada es UML[47]. Puntos 5.1.2 y 5.2.2.

4.3 Codificación

La codificación es la fase donde se implementa el código fuente. Puntos 5.1.3 y 5.2.3.

4.4 Pruebas

Las pruebas sirven para comprobar que el sistema funciona correctamente tras su codificación. Puntos 5.1.4 y 5.2.4.

4.5 Patrones de Diseño

4.5.1 Model-View-Controller

El patrón model-view-controller, conocido por sus siglas MVC, se suele utilizar para el desarrollo de aplicaciones con interfaz de usuario, ya que su principal objetivo es aislar la lógica del programa de la interfaz.

Como su nombre indica, este patrón se compone de tres elementos:

- **Modelo:** El modelo se encarga de la parte lógica del programa. Es el componente que estipula las reglas del sistema y maneja toda la información.
- **Vista:** La vista es la interfaz del sistema. Estipula la forma en que los datos han de ser mostrados al usuario.
- **Controlador:** El controlador es el intermediario entre la vista y el modelo. Recoge las acciones del usuario y las convierte en comandos o peticiones para enviárselas al modelo. A su vez, envía comandos a la vista para modificar la información que se muestra en pantalla cuando es modificada por el modelo.

4.5.2 Patrón Observador

El patrón observador proporciona una forma sencilla de notificación entre objetos. Cuando un objeto cambia su estado, notifica a todos los objetos dependientes del mismo.

El patrón está formado por cuatro objetos [48]:

- **Sujeto:** El sujeto proporciona una interfaz para agregar y eliminar observadores.
- **Observador:** El observador define el método que usa el sujeto para notificar los cambios.
- **Sujeto Concreto:** Se encarga de notificar a los observadores concretos.
- **Observador Concreto:** Mantiene una referencia al sujeto concreto e implementa la interfaz de actualización.

4.5.3 Patrón Fachada

El patrón fachada permite reducir la complejidad del sistema, introduciendo un objeto 'fachada' que actúa como interfaz entre dos partes del sistema, enmascarando una estructura más compleja.

La principal utilidad de dicho patrón es simplificar el sistema y permitir una fácil modificación futura del mismo, simplemente cambiando la implementación de la interfaz, o incluso tener distintas implementaciones para elegir cuál usar en cada momento.

Capítulo 5

Sistema

El objetivo principal del sistema es mejorar la impresión actual de imágenes médicas, ya que en algunos casos es difícil para los expertos corroborar un diagnóstico viendo la imagen o imágenes impresas. Por lo tanto, durante el desarrollo del sistema nos centraremos sobre todo en el análisis, ya que para lograr el objetivo es fundamental la investigación.

Este capítulo se dividirá en dos grandes secciones, que representan las dos iteraciones de desarrollo del sistema: la iteración 1, en la cual se investiga el proceso de impresión y se implementan las mejoras y la iteración 2, en la cual se diseña e implementa una nueva herramienta de calibración. Al principio de cada sección se explicará más en profundidad el objetivo y utilidad de cada iteración (ver puntos 5.1 y 5.2).

A su vez, cada iteración se dividirá en análisis, diseño, codificación y pruebas, explicados en el capítulo 4.

5.1 Primera iteración

El objetivo de esta primera iteración es lograr imprimir en papel las imágenes médicas con la mayor definición posible e intentando que se parezcan lo máximo posible a lo que un experto pueda ver en su pantalla especializada. Con la configuración actual, se puede apreciar claramente que la diferencia entre la imagen impresa y la mostrada en el monitor es elevada y, por lo tanto, es totalmente comprensible que al experto le cueste discernir los detalles de la imagen y, por tanto, malinterpretar un diagnóstico.

Aunque el proceso de impresión a primera vista pueda parecer un proceso simple, con las imágenes médicas no pasa lo mismo que con el resto de imágenes. La gente en general está acostumbrada a las imágenes típicas, con valores típicos y los colores típicos. Sin embargo, las imágenes médicas tienen unos rangos de valores muy elevados, y la necesidad de imprimir utilizando negro compuesto hace que las imágenes salgan con tonalidades de color, cuando en un principio son imágenes en escala de grises. Por ello, lo más importante de esta iteración

es comprender el proceso que sigue una imagen DICOM para llegar a ser impresa en papel e investigar por qué no salen como deberían salir, así como distintas formas de imprimirlas para lograr la calidad deseada.

5.1.1 Análisis

La diferente interpretación de la imagen por parte de cada uno de los elementos que forman parte del proceso de impresión puede provocar, y provoca, modificaciones no deseadas en las imágenes. Por ello, es importante analizar todo el proceso paso a paso.

Como proceso general, podemos dividirlo en tres pasos: La máquina o dispositivo médico que realiza el estudio genera la información propia de dicha modalidad (ver punto 2.1.1), y genera con ella un archivo DICOM; De dicho archivo se extrae la imagen médica, se guarda y se encapsula en un documento imprimible, generalmente un pdf; Dicho documento se manda imprimir, la impresora lo procesa y se imprime. En la figura 5.1 se puede observar el proceso general que sufre un archivo DICOM para ser impreso.

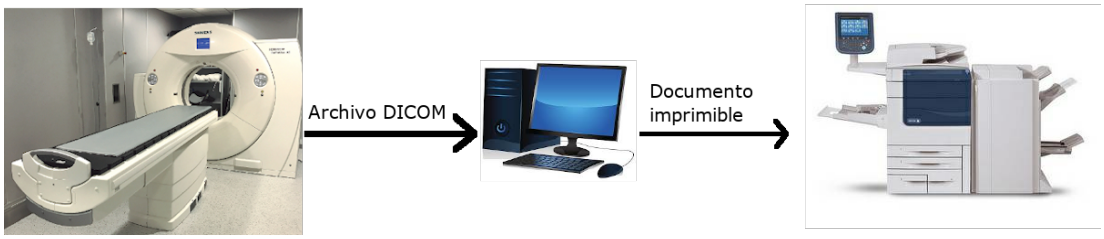


Figura 5.1: Proceso general

El objetivo de este análisis es encontrar la mejor manera de procesar el archivo DICOM e imprimirlo, por lo que nos centraremos sobre todo en la parte de la extracción y procesado de la imagen médica. A lo largo de este análisis usaremos la figura 3.2 usada en el capítulo 3 para irnos orientando y saber en todo momento qué parte del sistema estamos analizando y las pruebas que estamos realizando. Iremos modificando la imagen conforme progresemos en el análisis

Archivo DICOM

En lo que se refiere a la generación del archivo no se puede hacer nada, ya que eso depende de cada modalidad. Sin embargo, la extracción de la imagen médica sí que la realiza el sistema, y la forma de hacerlo determina en gran medida la calidad y detalle de los detalles de la imagen tras la impresión.

Para entender la forma en que se extrae la imagen, es fundamental familiarizarse primero con el estándar DICOM. Para ello, en la empresa me explicaron los fundamentos del estándar (ver punto 2.1.1), los tags más importantes (ver punto 2.1.1), el significado del Windowing

(ver punto 2.1.1), de la LUT (ver punto 2.1.1), me explicaron que actualmente para calibrar una imagen se usa el windowing, la LUT, el brillo y el contraste, y me indicaron la página en la que se encuentra el estándar actual [49].

Sin embargo, el impacto de los cambios en el windowing, la LUT, el brillo y el contraste no es fácil de comprender simplemente explicándolo, por lo que decidí investigarlo empíricamente. Para ello, usé el sistema ya existente para generar distintas imágenes en formato png cambiando dichos parámetros.

Para comenzar, y debido a que son conceptos más familiares, comencé modificando el brillo y el contraste. En la figura 5.2 se pueden observar cambios en el brillo y el contraste. Hacia la derecha aumenta el contraste y hacia arriba aumenta el brillo. Como se puede apreciar, las imágenes con mayor contraste parecen ”quemadas”, es decir, las zonas blancas se ven demasiado blancas; sin embargo, en las imágenes con menor contraste es más difícil apreciar los detalles. Por otro lado, con el brillo pasa algo similar. Al aumentar el brillo se ven mejor

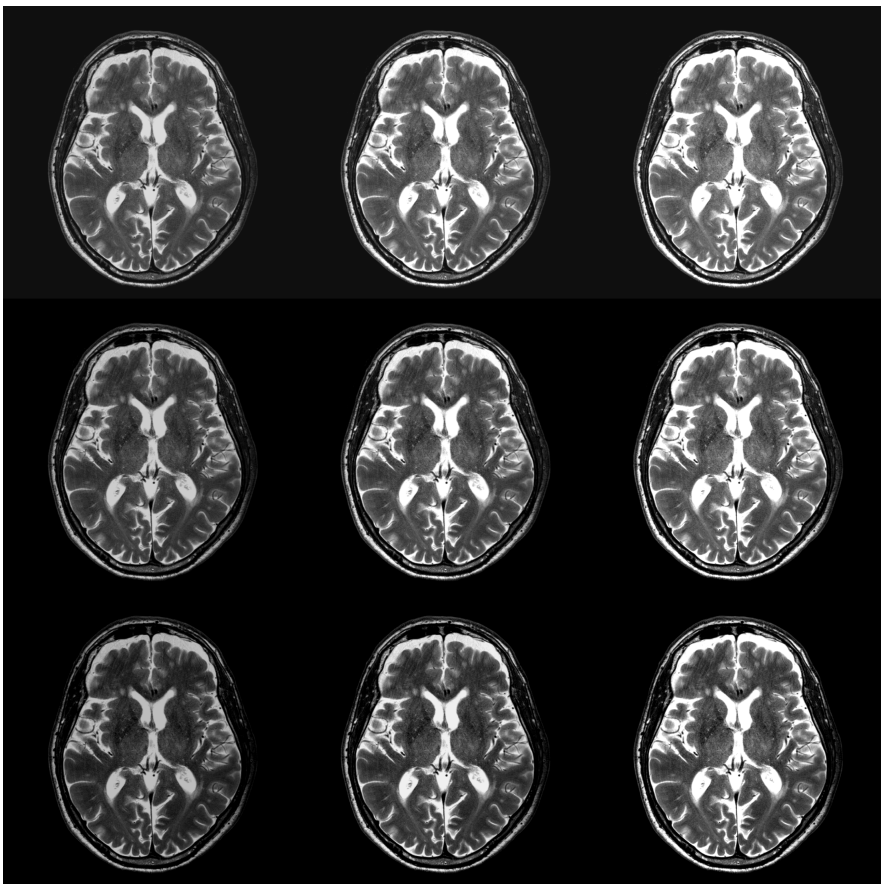


Figura 5.2: Cambios de brillo y contraste

los detalles de las zonas oscuras, pero se pierden los detalles en las zonas claras. De forma inversa, al disminuir el brillo se aprecian mejor los detalles en las zonas claras pero se pierden

en las zonas oscuras. Cabe destacar que dependiendo del formato de impresión quizá no se aprecien los cambios correctamente.

Tras entender el impacto del brillo y el contraste, llegó el turno del windowing. Para entenderlo mejor, es aconsejable entender primero lo que es el histograma de una imagen, explicado en el punto 2.1.3.

Como ya se ha explicado en el punto 2.1.1, el windowing define la ventana del histograma que se interpolará a valores de gris entre 0 y 255. Está compuesto por dos parámetros: window width, que define el ancho de la ventana y window center, que define el centro de la ventana (ambos valores son valores en el pixel data). A continuación se explicará con ejemplos gráficos cómo afectan los cambios en el windowing a las imágenes médicas.

En la figura 5.3 se muestra una imagen con un pixel data con valores entre 0 y 4000 y le aplicamos unos valores de windowing que permiten mostrar todos los valores del histograma (window center 2000 y window width 4000). La tomaremos de referencia para explicar los siguientes cambios.

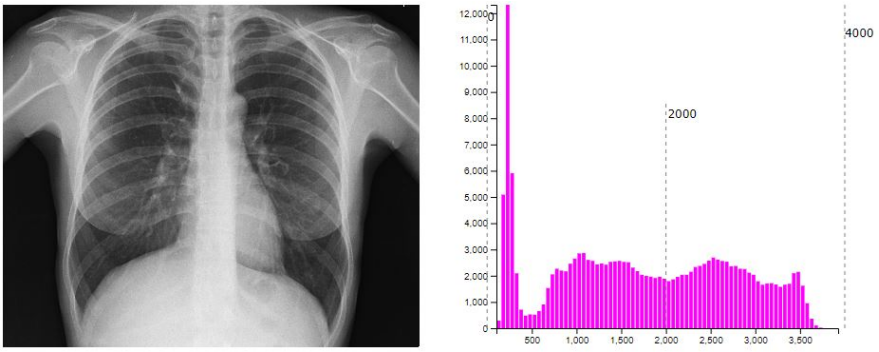


Figura 5.3: Imagen médica con máximo windowing

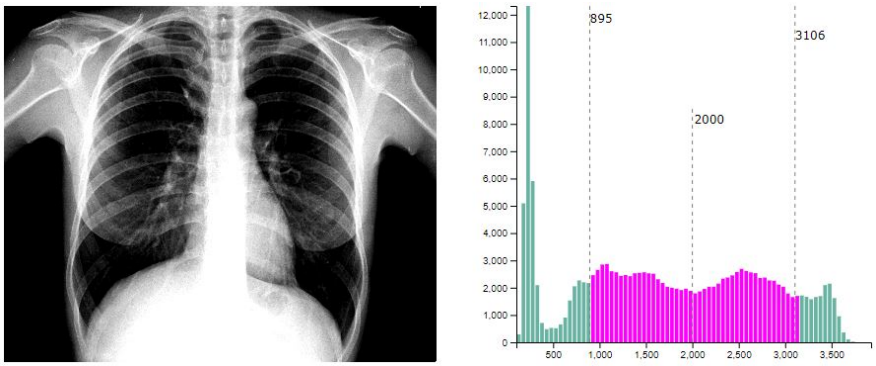


Figura 5.4: Imagen médica disminuyendo el valor del window width

En la figura 5.4, se ha disminuido el window width con respecto a nuestra imagen de referencia. Como se puede observar, tanto los valores más oscuros como los más claros han

pasado a ser valores extremos, mientras que los valores intermedios ahora tienen mayor rango dinámico, por lo que se pueden apreciar mejor los detalles.

En la figura 5.5 se han disminuido tanto el window width como el window center, y como se puede observar, las zonas claras han adoptado todas el valor máximo de blanco; sin embargo, en el resto de la imagen se pueden apreciar mucho mejor los detalles, debido a que se ha aumentado su rango dinámico, es decir, ahora existen muchos más valores para interpretar los mismos píxeles. Mientras antes solamente existían, por ejemplo, 100 valores de gris para interpretar 1000 valores del Pixel Data, ahora existen 255. Como es de suponer, si se aumenta el window center ocurre el caso inverso, como se puede ver en la figura 5.6.

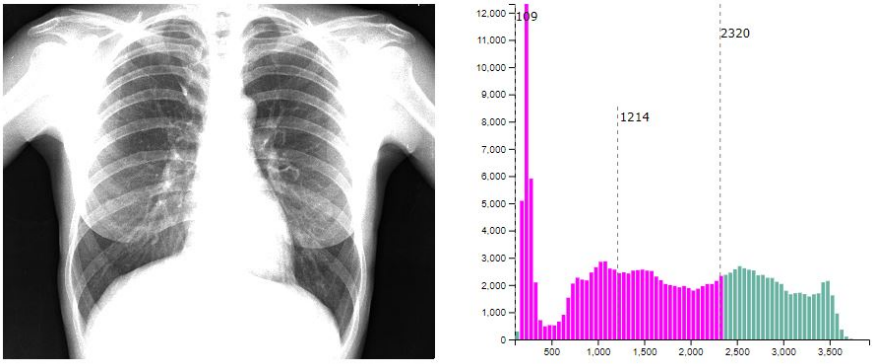


Figura 5.5: Imagen médica disminuyendo el valor del window width y del window center

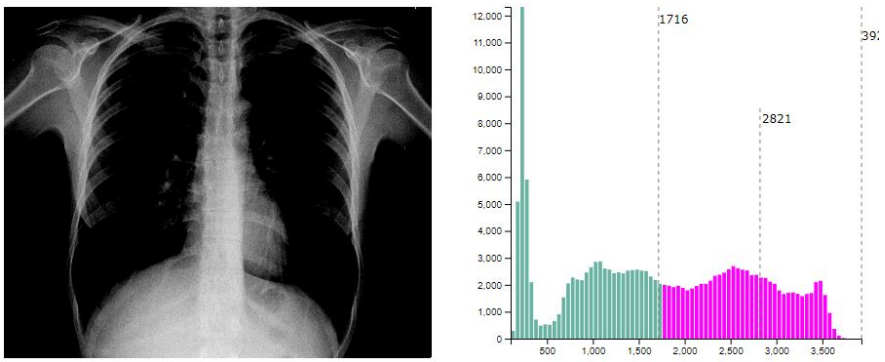


Figura 5.6: Imagen médica disminuyendo el valor del window width y aumentando el del window center

La LUT complementa al windowing, y nos permite modificar la forma en que se interpola el pixel data de la imagen DICOM para pasarla a valores que entienda el ordenador (0 a 255). Para poder entender mejor el impacto de la modificación de la LUT, usé una herramienta en el sistema ya existente que permitía modificarla a través de una función. En la figura 5.7 se puede ver un gráfico de una función LUT sin modificar, junto con la imagen de prueba. En el eje X

se encuentran los valores del pixel data y en el eje Y los valores de la interpolación (0 a 255). Como podemos observar en el gráfico, los valores más a la izquierda se corresponden con los más oscuros, mientras que los más a la derecha se corresponden con los valores más claros. Por lo tanto, podemos suponer que si cambiamos solamente parte de la línea recta, estaremos modificando únicamente esos valores de gris. Además, podemos ver que los valores en el eje X van desde el 0 al 4000, por lo que podemos suponer que el window width es 4000 y el window center es 2000.

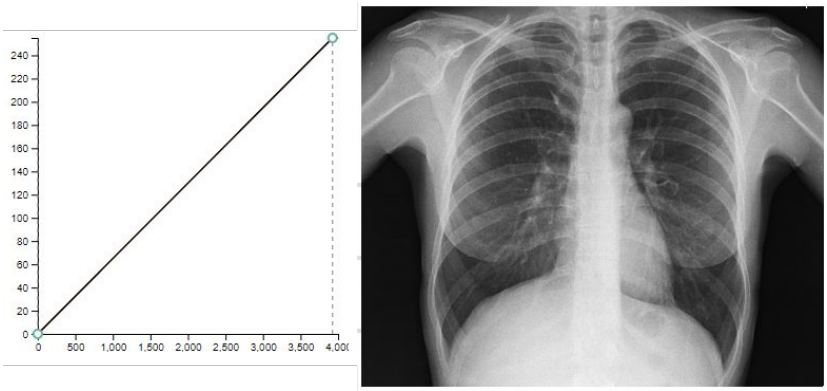


Figura 5.7: Imagen médica con LUT por defecto

Como podemos observar en la figura 5.8, aumentando la pendiente de una zona de la función, se aumenta el contraste entre dichos valores de gris, mientras que disminuyendo la pendiente se produce el efecto contrario. En esta imagen podemos ver cómo en las zonas internas del tórax se aprecia un mayor contraste, permitiendo diferenciar con mayor facilidad los detalles. Sin embargo, tanto en la zona de la columna vertebral como en el diafragma se ha producido una disminución del contraste, por lo que resulta más complicado ver detalles en esas zonas.

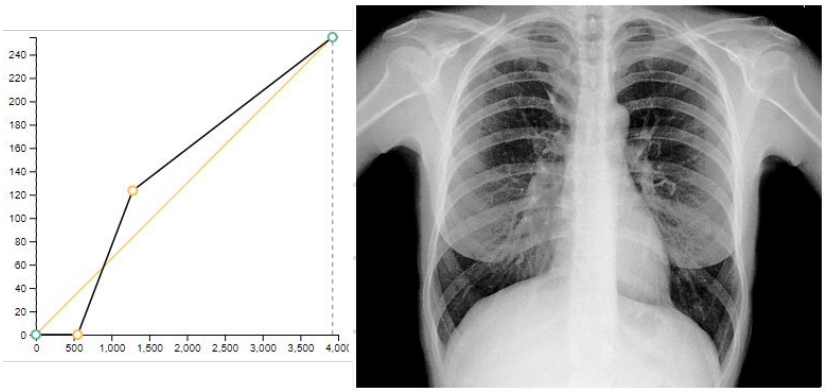


Figura 5.8: Imagen médica con aumento de contraste en Zona LUT

Por último, en la figura 5.9 podemos ver que moviendo en vertical un punto en concreto de la función, aumenta el brillo de los valores correspondientes de gris. En la zona interna del tórax podemos ver que todos los valores de gris que se encuentran dentro del rango de la Zona LUT, ahora tienen mucho más brillo. Este ejemplo es exagerado para poder apreciar bien el cambio, por lo que no se ve reflejada su utilidad real.

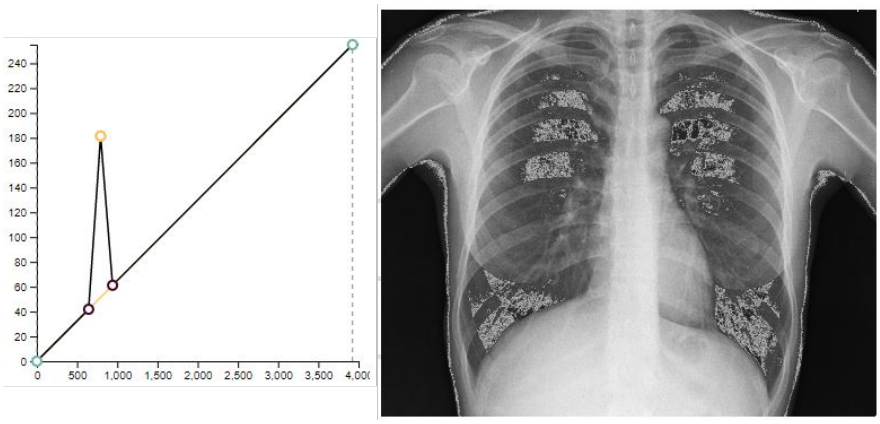


Figura 5.9: Imagen médica con aumento de brillo en Zona LUT

Consideraciones El uso del windowing es fundamental para acotar los niveles de gris que se quieren imprimir, para así aumentar el rango dinámico de la imagen y permitir observar mejor todos los detalles. Además, hacer ciertos cambios en la LUT permite destacar aquellos niveles de gris pertenecientes a ciertas zonas cruciales de la imagen, para así poder apreciar mucho mejor los detalles en esas zonas. Para poder hacer uso de ello, habría que crear un sistema que modifique el windowing y la LUT de la imagen DICOM para que se muestre correctamente tras la impresión. Una idea podría ser un sistema inteligente capaz de interpretar el pixel data y decirnos el windowing y la LUT ideales (ver punto 7.4). Otra idea interesante sería modificar la herramienta actual que se utiliza para comunicarse con el sistema para permitir modificar el windowing y la LUT de una manera mucho más gráfica, simple e intuitiva.

Además, puedo concluir que es innecesario modificar el brillo o contraste general de la imagen una vez convertida al png, como se hacía en el sistema ya existente, ya que dicha función se puede realizar con la LUT de una forma más precisa. Además, al ser cambios hechos sobre la imagen ya transformada a png (valores entre 0 y 255), cambiar el brillo o el contraste produce pérdidas de información.

Toma de contacto con las impresoras

Para continuar familiarizándome con el sistema ya existente, mi siguiente paso fue imprimir algunas imágenes de prueba para comprender de primera mano las diferencias entre

las imágenes impresas y las que se muestran en pantalla, además de las diferencias entre los dos métodos de impresión que tenían implementados (utilizando Adobe Acrobat Reader y utilizando la librería PdfBox). Me comentaron que tienen dos métodos de impresión distintos porque cada uno tiene unas propiedades características, y además de que cada impresora hace una interpretación distinta, cada cliente tiene sus propias preferencias. Tras imprimir de cada una de las formas, se puede comprobar claramente que existe una diferencia entre ellas y sobretodo con la imagen en pantalla. Las imágenes impresas con Adobe Acrobat Reader se ven más oscuras, mientras que las impresas con PdfBox parece que tiran hacia un color que, dependiendo de la luz ambiente, se podría interpretar como un color sepia o como un color verdoso. Esta diferencia de color se debe a que las imágenes son impresas con una propiedad llamada negro compuesto. Esto significa que la impresora, en lugar de utilizar la tinta negra, o en este caso tóner, mezcla los otros tres colores. Esto es debido a dos motivos: primero, en el mismo documento en el que se imprimen las imágenes en escala de grises, se imprimen una portada y una contraportada que van a color, por lo tanto, si se establece que la impresora debe imprimir en escala de grises, dichas portada y contraportada se no se imprimirían a color. Además, la creación de los grises con cian, magenta y amarillo proporciona una cobertura sobre el papel que genera una impresión visual mucho mejor, por no mencionar la profundidad del fondo negro de las imágenes.

En cambio, las diferencias entre la imagen impresa y la que se muestra en pantalla son mayores. No solo se aprecia una diferencia en el color, sino que también se aprecia un gran cambio en la calidad de los detalles. Para solucionar las diferencias de color traté de calibrar el monitor en el que visualizaba las imágenes, pero existe una gran diferencia entre un monitor y un papel que impide que nuestro ojo vea las dos imágenes iguales, y es que el monitor emite luz, mientras que el papel la refleja. Por lo tanto, mientras que los colores del monitor se mantienen iguales bajo cualquier ambiente, los colores impresos adquieren distintas tonalidades dependiendo de la luz que reflejen, y aunque en las pruebas realizadas con la impresora Xerox Color 550 el negro compuesto estaba muy conseguido y apenas se apreciaban reflejos de ningún color, en la impresora Xerox C8035 y en la Xerox Phaser 7760 estos reflejos eran mucho más notorios, por lo que empecé a investigar sobre las impresoras a mi disposición para conocer el por qué de esas diferencias en la impresión.

Durante mi investigación sobre las impresoras, me topé con que son de las mejores del mercado, con resoluciones de hasta 2400 puntos por pulgada (ppp) y de los mejores procesadores de color, lo cual hace a las tres idóneas para la impresión de estas imágenes; sin embargo, la Xerox Color 550 es de una gama más elevada, seguida por la C8035 y por último la Phaser 7760.

Consideraciones: La conclusión es que cuanto mayor sea la gama de la impresora, mejor imprimirá. No obstante, me resultó extraño que se notase una clara diferencia de resolución entre la imagen que se ve en pantalla con respecto a la impresa, ya que imprimiendo con una resolución de 2400 ppp debería verse incluso con mayor definición que en el monitor, si la imagen es lo suficientemente grande. Aquí comencé a buscar el punto en el proceso de impresión que más influyese en ese decremento de resolución, y aplicando un poco la lógica, supuse que uno de los pasos que más podría influir sería el de generar un archivo png con todas las imágenes del estudio.

Inserción de las imágenes en el pdf plantilla

Dicho archivo png se generaba para colocar las imágenes en una cierta disposición antes de insertarlas en el pdf, y aunque la disposición depende del cliente, suele ser un 2x3 (dos columnas y tres filas). Debido a que la disposición de las imágenes suele ser en forma de rejilla, a partir de ahora le llamaremos ”grid” (explicado en el punto 3.1).

Lo que llamamos pdf plantilla, no es más que un formulario pdf con un campo ”imagen” que ocupa toda la página. Dicho pdf se usaba para insertar el grid de imágenes en el campo ”imagen”, de esta forma se puede crear un pdf con el grid de imágenes de una forma sencilla y sin necesidad de que el sistema cree un pdf de cero.

La parte del sistema que estamos analizando actualmente se resalta en la figura 5.10.

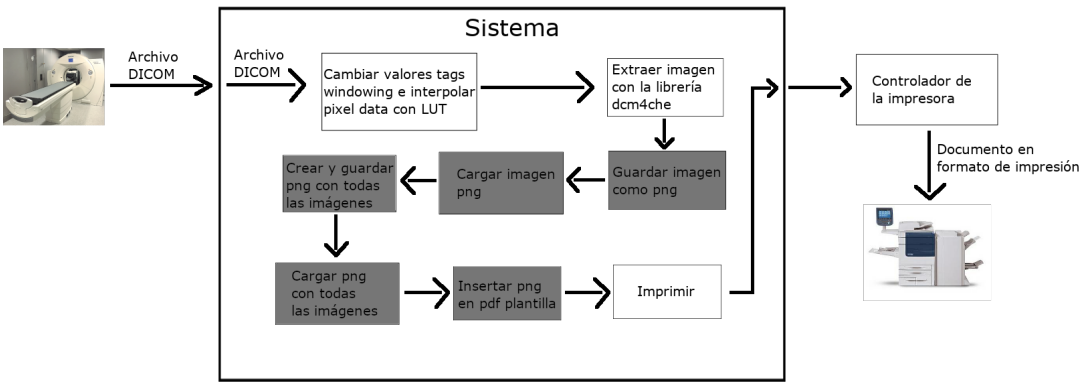
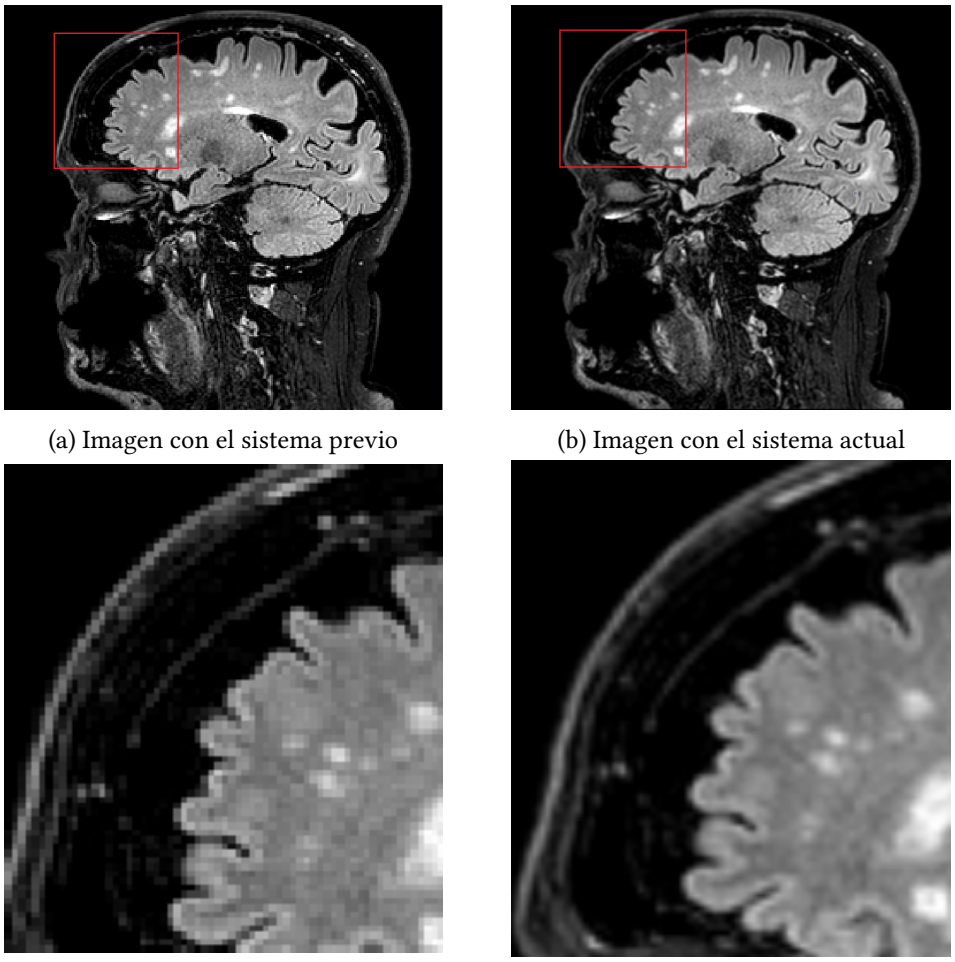


Figura 5.10: Estructura del sistema

Para saber si realmente la generación del grid estaba afectando al decremento de resolución de las imágenes, comencé a estudiar el código fuente del sistema existente, y en efecto, las imágenes eran escaladas antes de la generación del grid, para que el archivo png tuviese el tamaño de una hoja A4. Por lo tanto, la primera aproximación para mejorar la resolución de las imágenes impresas fue omitir el paso de generación del grid. Para omitir este paso, busqué una forma de crear un pdf como el que se usaba de plantilla (ver punto 3.1), es decir,

un formulario, que contenga seis imágenes (dos columnas y tres filas). Para ello, utilicé la herramienta gratuita LibreOffice Draw y modifiqué el sistema para introducir las imágenes png directamente en el pdf, sin pasar por escalarlas y unirlas todas en un mismo png.

Tras imprimir los pdfs creados y comparándolos con los anteriores, se podía apreciar claramente una mayor nitidez, sobretodo imprimiendo las imágenes en un mayor tamaño. Esto es debido a que durante la creación del grid como archivo png, las imágenes se escalaban a una menor resolución, por lo que se perdía información al insertarlas en el pdf. Sin embargo, al incluir las imágenes directamente en el pdf, éstas se encapsulan y no pierden definición.



(a) Imagen con el sistema previo

(b) Imagen con el sistema actual

(c) Imagen aumentada con el sistema previo

(d) Imagen aumentada con el sistema actual

Figura 5.11: Diferencias de nitidez entre las imágenes

En las figuras 5.11 se puede apreciar cómo la imagen de la figura 5.11c (sistema anterior) se ve más pixelada que la imagen de la figura 5.11d. Dichas imágenes son sacadas directamente de los pdf antes de imprimirlos. A pesar de que pueda no parecer muy relevante, en papel

esa diferencia de nitidez permite apreciar mucho mejor los detalles, sobretodo a ojos de un experto.

Para asegurar que la pérdida de calidad no se produce por guardar una imagen como png, también modifiqué la inserción de la imagen en el pdf para poder insertarla directamente sin pasar por el paso de guardarla como png. Aunque la diferencia no se aprecia tras imprimir las imágenes, es un paso que podemos obviar ya que el único propósito de su existencia era la generación del grid png.

Consideraciones: Hay que tratar de evitar el escalado de las imágenes y minimizar los cambios de formato, ya que pueden producir pérdidas de información no deseadas. Tras esta parte del análisis, la estructura del sistema quedaría como se puede ver en la figura 5.12

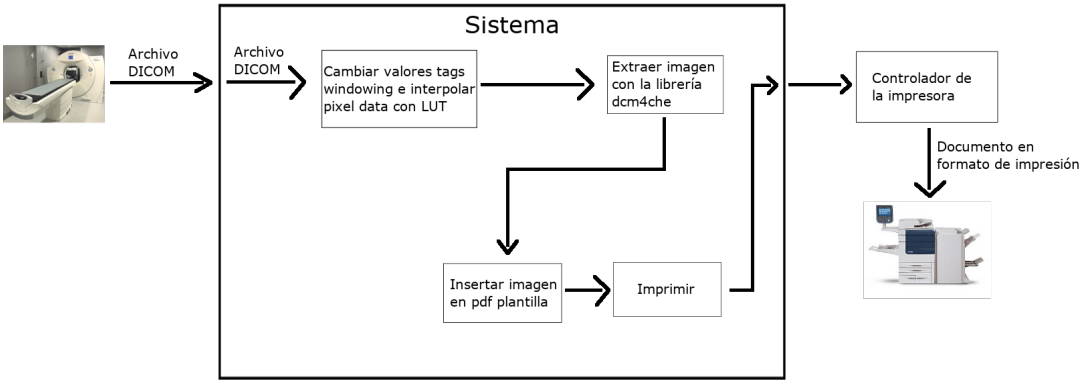


Figura 5.12: Diagrama tras suprimir la generación del grid

Creación del pdf plantilla

El siguiente paso fue buscar distintas opciones para generar el pdf plantilla, ya que hasta ahora utilizaba el programa LibreOffice Draw para generar los formularios y la librería PdfBox para rellenarlos con las imágenes.

Uno de los principales problemas de crear los formularios de esta forma, es que se requiere la existencia de un formulario por cada posible disposición de las imágenes que requiera el cliente, por lo que la primera idea lógica era crear el pdf plantilla interactivamente, es decir, el programa generaría el pdf plantilla automáticamente bajo unos criterios dados.

Crear el pdf plantilla interactivamente fue muy costoso en tiempo, ya que requería buscar mucha información sobre cómo se generan los pdf y cómo se tratan los elementos del mismo. En un primer momento utilicé la librería PdfBox para generar el pdf, generando primero un pdf en blanco y calculando el rectángulo que debe ocupar cada imagen en función del tamaño de la hoja y la disposición de las imágenes.

Debido a que ahora el sistema también crea el pdf plantilla, el flujo del programa quedaría como se muestra en la figura 5.13.

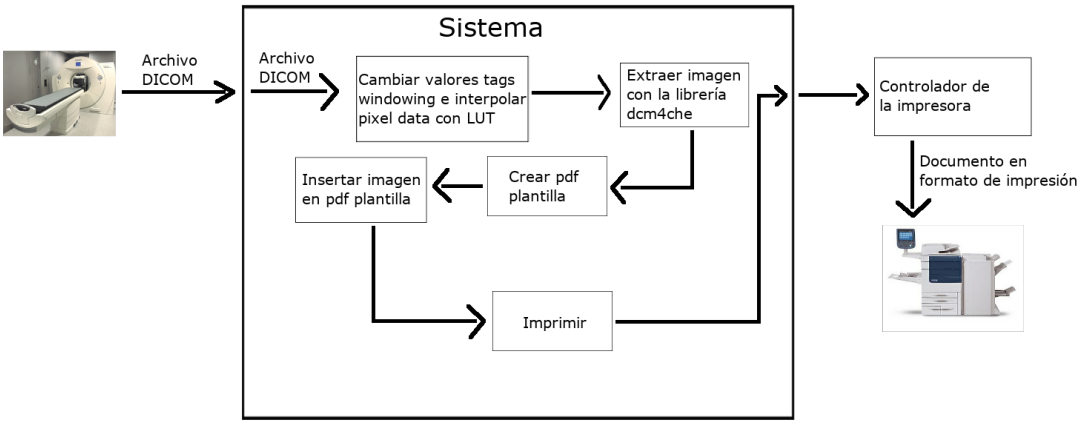


Figura 5.13: Diagrama tras implementar la creación del pdf plantilla

Sin embargo, una de las cosas que se me propuso para intentar mejorar la impresión del pdf fue buscar otra nueva librería que me permitiese tanto insertar las imágenes en el pdf como imprimirlo e investigar si existe alguna mejora. Comencé a buscar y me topé con dos nuevas posibilidades: IText y AdobePDFLibrary, y debido a que el estándar pdf fue desarrollado por Adobe, decidí probar con la librería AdobePDFLibrary.

Toma de contacto con la librería AdobePDFLibrary

Durante la búsqueda de todas las posibilidades que ofrece esta librería, encontré algunas muy interesantes: Convertir el espacio de color de un documento pdf, crear un pdf en formato PDF/X y transformar un PDF/A (formato que se utiliza habitualmente) en PDF/X. El formato PDF/X es un estándar pensado para imprimir en papel, ya que posee ciertos requisitos para mejorar la impresión. Por ejemplo, en el estándar PDF/X-1a se estipula que todas las imágenes han de ir en espacio de color CMYK (el usado por la mayoría de las impresoras). Y además, según la documentación de la librería, durante la transformación de PDF/A en PDF/X se realizarían las transformaciones necesarias para que el documento cumpla el estándar.

Lo primero que hice fue probar a convertir un pdf creado anteriormente en un pdf en formato PDF/X, a través de la librería AdobePDFLibrary e imprimirlo con las opciones de impresión por defecto. Por desgracia, las imágenes se imprimían con una tonalidad rosada y muy oscuras, lo cual dificultaba comparar la imagen en pantalla con la impresa. Aunque este formato quizá fuese prometedor, lo descarté por el momento debido a los problemas ya dichos y ciertos problemas de compatibilidad de la librería de Adobe. Esta conversión se situaría en la parte del sistema resaltada en la figura 5.14.

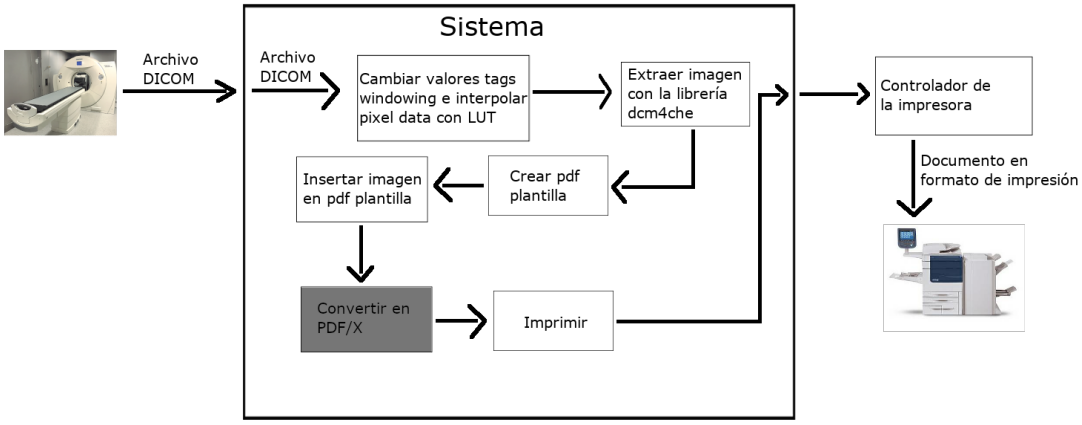


Figura 5.14: Diagrama tras implementar la conversión en PDF/X

Tras emplear de nuevo una cierta cantidad de tiempo desarrollando una forma de rellenar el pdf plantilla con AdobePDFLibrary, guardarlo e imprimirlo, concluí que las diferencias entre insertar las imágenes con esta librería o con PdfBox no son apreciables; sin embargo, al buscar información sobre cómo imprimir el pdf con la librería de Adobe, encontré multitud de parámetros de impresión, por lo que continué la investigación por este camino.

La estructura del programa tras las modificaciones realizadas se muestran en la figura 5.15, donde a su vez se puede ver resaltada la zona del sistema donde se han realizado las modificaciones para insertar las imágenes utilizando la librería de Adobe.

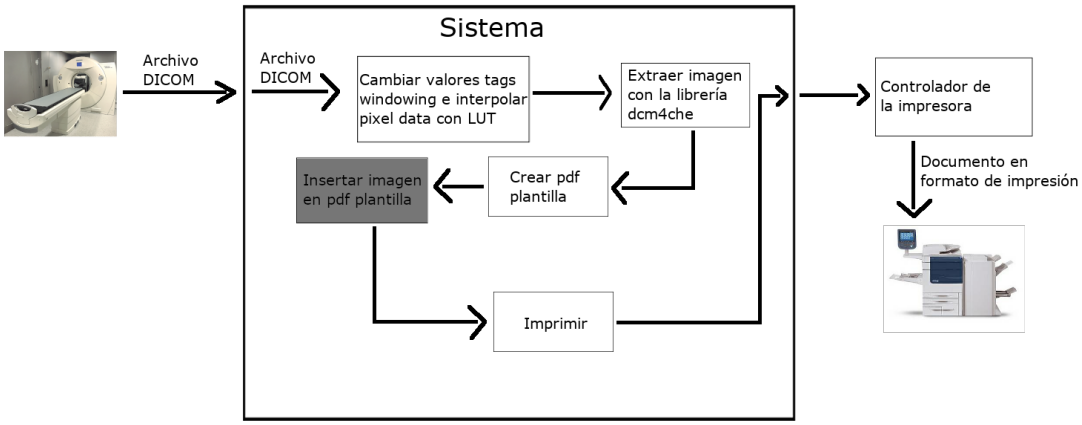


Figura 5.15: Diagrama tras implementar la creación del pdf plantilla

Parámetros de impresión

En la figura 5.16 se resalta la parte del sistema que se tratará en este apartado.

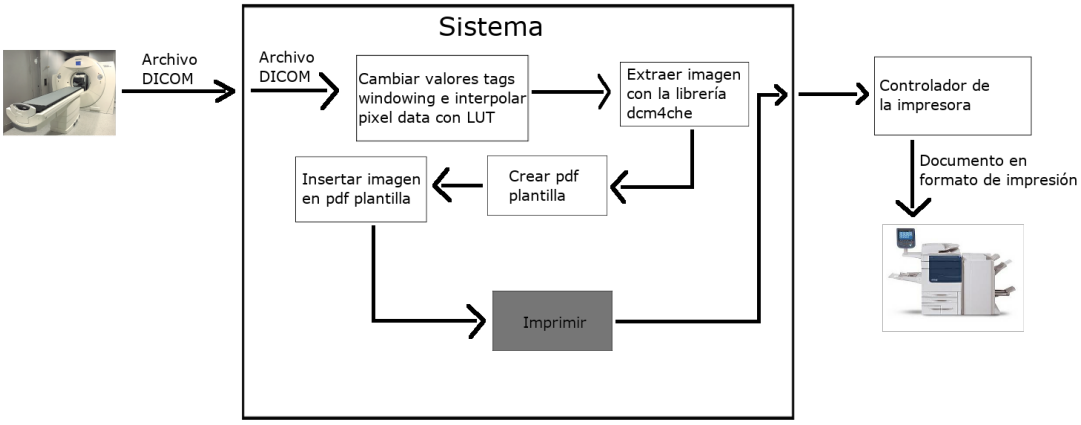


Figura 5.16: Diagrama de la estructura del programa

Para imprimir correctamente un trabajo es fundamental conocer los parámetros de impresión que proporcionan las librerías y modificarlos en función de la necesidad. En este caso, dispongo de tres formas de imprimir un pdf: Con la librería AdobePDFLibrary, con la clase PrinterJob del paquete java.awt.print y con Adobe Reader. Desgraciadamente, solo disponía de distintos parámetros de impresión con AdobePDFLibrary y con PrinterJob:

- **AdobePDFLibrary:** Esta librería provee una gran cantidad de opciones y parámetros de impresión. Tras investigar la utilidad de cada uno, he aislado los que podrían ser relevantes para la tarea que nos ocupa:
  - **setBitmapResolution:** puntos por pulgada para los mapas de bits. El valor por defecto es 300. Este parámetro es muy interesante, ya que las impresoras utilizadas admiten hasta una resolución de 2400 puntos por pulgada (ver punto 2.2.10). No puede ser usado cuando el parámetro setUseFullResolutionJP2KData tiene valor ”true”.
  - **setDisableFlattening:** Elimina las transparencias en la imagen. Quizá a primera vista no parece un parámetro importante, pero vale la pena probarlo porque no sabemos lo que hacen las librerías al interpretar las imágenes en el pdf.
  - **setEmitBG:** La definición de este parámetro es ”Emit black generation” (emitir generación de negro). Debido a que la definición es escueta y no dice qué hace en concreto, decidí probar las dos opciones de este parámetro (true y false).
  - **setEmitDeviceExtGState:** Cuando el valor de este parámetro es ”true” (valor por defecto), al generar los gráficos del pdf para imprimirlo, se incluyen los parámetros dependientes del dispositivo (como por ejemplo generación de negro, negro compuesto, halftones, etc.). Consideré importante este parámetro porque quizá al

dejar la interpretación del pdf en manos únicamente del controlador, la generación de las imágenes cambiaba.

- **setEmitHalftones:** Emitir halftones. Los halftones, o semitonos, son una técnica de impresión que consiste en simular una imagen con tonos continuos mediante el efecto óptico producido por una serie de puntos de un mismo color, simplemente variando el espaciado entre ellos. En un principio consideré que merecía la pena probar esta característica, pero buscando más información sobre la librería descubrí que el parámetro solo es efectivo cuando dichos halftones se encuentran ya presentes en el pdf.
- **setEmitRawData:** Al poner el valor de este parámetro como ”true”, evita poner filtros innecesarios en la imagen. El valor por defecto de este parámetro es false.
- **setEmitUCR:** UCR son las siglas de Under Color Removal. Al poner el valor de este parámetro como ”true”, elimina el exceso de cian, magenta y amarillo que se habrían añadido para formar negro, para reemplazarlos por tinta negra. Por defecto el valor de este parámetro es ”true”. Me interesa mantener el valor de este parámetro como ”false”, ya que lo que quiero es utilizar negro compuesto (explicado en el punto 2.2.10).
- **setGradientResolution:** Puntos por pulgada de los degradados. Por defecto el valor es 150. Como ya se ha destacado en setBitmapResolution, las impresoras admiten una definición de 2400 puntos por pulgada. No puede ser usado cuando el parámetro setUseFullResolutionJP2KData tiene valor ”true”.
- **setHostBasedCM:** Hacer la manipulación de los colores usando el perfil del dispositivo desde el que se imprime. El valor por defecto es ”false”.
- **setOptimizeForSpeed:** Como su nombre indica, optimiza el documento para una mayor velocidad de impresión. Aunque en la documentación especifica que los cambios efectuados se hacen sobre las fuentes del documento, consideré que valía la pena probar a cambiar su valor. El valor por defecto es ”true”.
- **setSuppressCSA:** No emite ”CSAs” para los colores de 4 componentes (CMYK). CSA se refiere al perfil de color que tiene el documento generado con AdobePDFLibrary. El valor por defecto es ”false”.
- **setTransparecyQuality:** Establece la calidad con la que se imprimen las transparencias. Los valores van del 0 al 100 y el valor por defecto es 80.
- **setUseFullResolutionJP2KData:** Determina si se debe usar la máxima resolución disponible del estándar JPEG2000. El valor por defecto es false.

- **PrinterJob:** El paquete javax.print.attribute cuenta con algunas clases para estipular ciertos parámetros de impresión, como por ejemplo el tamaño de la hoja o la resolución

que debe usar la impresora. Por desgracia, solo encontré un atributo que realmente permitiese cambiar la resolución del trabajo a imprimir (javax.print.attribute.standard.PrintQuality), aunque por defecto ya se selecciona la mejor calidad posible.

A esta altura del desarrollo dejé de tener a mi disposición la impresora Xerox Color 550, que era la que mejores resultados estaba dando. Y debido a que las diferencias de las impresiones entre la Xerox C8035 y la Xerox Phaser 7760 eran inapreciables, por comodidad continué las pruebas únicamente con la Xerox C8035.

Los resultados de aplicar cambios en los parámetros de impresión de la librería de Adobe fueron los siguientes:

- **setBitmapResolution:** Al establecer el valor en 2400, se aprecia una ligera mejora de resolución en la imagen.
- **setDisableFlattening:** No parece que haya ninguna diferencia apreciable, lo cual no era de extrañar.
- **setEmitBG:** Tampoco parece que este parámetro produzca ningún efecto sobre la imagen impresa.
- **setEmitDeviceExtGState:** Tampoco produce ningún cambio, lo cual da a entender que la librería no produce ninguna modificación en el pdf al imprimirlo, a parte de las indicadas los parámetros de impresión.
- **setEmitHalftones:** Confirmando la información encontrada mencionada anteriormente, este parámetro no genera semitonos si no están ya presentes en el pdf.
- **setEmitRawData:** No se aprecia ninguna diferencia a simple vista, por lo que entiendo que de por sí ya no se aplicaba ningún filtro innecesario
- **setEmitUCR:** Tras probarlo, la conclusión es que este parámetro simplemente alterna entre negro compuesto y negro puro, por lo que no me interesa cambiarlo. Cabe destacar que la opción del controlador sobreescribe este parámetro
- **setGradientResolution:** Al establecer el valor en 2400, se aprecia una ligera mejora de resolución en algunas zonas de la imagen. Al usarlo en combinación con el parámetro setBitmapResolution, se aprecian los mejores resultados.
- **setHostBasedCM:** Este parámetro produce que la imagen salga con un color mucho más rosado, por lo que no nos interesa.
- **setOptimizeForSpeed:** No parece que afecte a la imagen impresa.

- **setSuppressCSA:** La imagen sale con un tono rosado, aunque menos que la opción setHostBasedCM. No nos interesa
- **setTransparecyQuality:** Al no surtir efecto el cambio del parámetro setDisableFlattening, suponemos que no hay transparencias en el pdf, o al menos no influyen en su impresión, por lo que este parámetro no tiene efecto.
- **setUseFullResolutionJP2KData:** Se aprecia una mejor resolución de las imágenes que con los parámetros por defecto, más o menos a la altura de las pruebas con los parámetros setBitmapResolution y setGradientResolution; sin embargo las imágenes salen con un tono algo más rosado.

Consideraciones: Para la impresora utilizada y en el entorno en el que me encontraba, los únicos cambios que debo realizar en los parámetros para conseguir el mejor resultado son cambiar los valores de BitmapResolution y GradientResolution a 2400 y dejar el resto de parámetros por defecto; sin embargo, es aconsejable probar distintos parámetros, ya que dependiendo de la impresora y del entorno los resultados pueden variar.

Espacios de color con la librería AdobePDFLibrary

Tras hacer pruebas con los parámetros de impresión, era momento de hacer pruebas con los cambios en el espacio de color. Como ya comenté anteriormente, una de las opciones más interesantes que ofrece AdobePDFLibrary es cambiar el espacio de color del documento, permitiendo así cambiar el perfil de color de todo el documento y, por tanto, la interpretación del color que realizará más tarde la impresora. Existen dos métodos complementarios que permiten elegir el espacio de color en el que se transformará el documento: setConvertIntent y setConvertProfile. Podríamos decir que setConvertProfile determina el perfil de color a usar y setConvertIntent determina la forma en la que se aplicará dicho perfil.

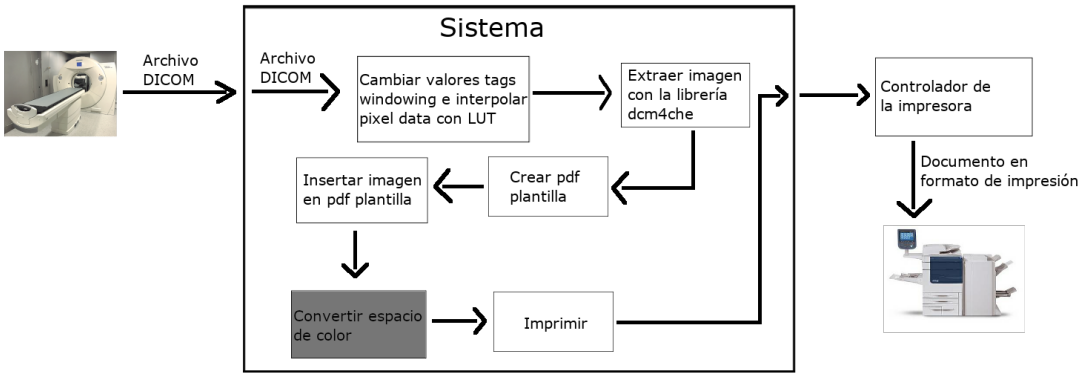


Figura 5.17: Diagrama de la estructura del programa

En la figura 5.17 podemos ver la parte del sistema que se estudia en este apartado. Como podemos ver, se añade un paso para cambiar el espacio de color del documento tras añadir las imágenes.

Dentro de las formas en que se puede aplicar el perfil tenemos distintas opciones:

- **Abs_Colorimetric:** Intenta mantener la exactitud del color a cambio de la relación entre colores. Por ejemplo, si en el espacio de color origen tenemos 255 valores para un cierto componente y en el espacio de color destino existen solamente 50, muchos colores que se ven distintos en el espacio de color origen se verán igual en el destino.
- **Rel_Colorimetric:** Hace lo mismo que Abs_Colorimetric pero compara el ”white point” del espacio de color origen con el de destino y realiza una interpolación para todo el resto de colores. El ”white point” es el valor que toman los componentes de un espacio de color para representar el color blanco.
- **Perceptual:** Intenta preservar la relación entre los colores tal y como la interpreta el ojo humano, aunque algunos colores deban cambiar.
- **Saturation:** Intenta crear colores vívidos a cambio de exactitud de los colores. Escala el espacio de color origen para preservar la relación relativa de saturación entre los colores. Es útil, por ejemplo, en gráficos, dónde no importa tanto la relación entre los colores como que se note la diferencia entre los mismos y estén brillantes y saturados.

La librería de Adobe nos provee 17 perfiles de color para usar con el método setConvertProfile. Entre ellos se encuentran perfiles que nos permiten cambiar el espacio de color a CMYK, XYZ, RGB y hasta escalas de grises.

Para comprobar cuáles son las mejores opciones, tanto el perfil de color como la forma de aplicarlo, primero he probado todos los perfiles de color, dejando el valor por defecto de setConvertIntent, que es Perceptual.

- **Acrobat CMYK:** Hay dos perfiles de color de este tipo: Actobat_5_CMYK y Acrobat_9_CMYK. Ambos son perfiles de color que permiten convertir el espacio de color a CMYK, que es el espacio de color que utilizan las impresoras. A pesar de ello, ambos perfiles imprimen la imagen con un tono más rosado y sin diferencias apreciables entre ellos.
- **Adobe RGB, Apple RGB y Color Match RGB:** Los tres son perfiles de color con componentes RGB e imprimen unos resultados prácticamente idénticos. No hay diferencia en comparación con lo ya impreso.
- **Dot Gain:** Hay cinco perfiles de color de este tipo: DOT_GAIN_10, DOT_GAIN_15, DOT_GAIN_20, DOT_GAIN_25 y DOT_GAIN_30. Todos ellos son perfiles de color de

una sola componente, por lo que son perfiles en escala de grises. El número simplemente significa la ganancia de punto que debe emular. Debido a que la ganancia de punto es algo que precisamente queremos evitar, no nos valen estos perfiles.

- **Flat XYZ y PCS XYZ:** Dos perfiles de color con tres componentes XYZ. La imagen impresa tiene colores no deseados, debido a que el espacio de color XYZ no es el indicado ni para monitores ni para impresoras.
- **Gamma:** Existen dos perfiles de color de este tipo: GAMMA_18 y GAMMA_22. Son perfiles de color con un espacio en escala de grises, pero el resultado es muy brillante con ambos perfiles, de ahí su nombre.
- **LAB:** Solo hay un perfil con el espacio de color LAB: LAB_D_50. Al igual que con el espacio XYZ, el resultado no es satisfactorio ya que la imagen impresa tiene colores no desesados.
- **Monitor RGB y sRGB:** Ambos perfiles de color son el mismo, ya que el monitor utilizado usaba el perfil sRGB. A pesar de cambiar el espacio de color del documento a uno RGB, es el perfil con el que mejores resultados obtuve.

Tras probar los perfiles, llegó el turno de probar las distintas formas de aplicar los perfiles. Para ello, usé los perfiles Acrobat_9_CMYK, ya que proporciona un espacio de color CMYK, y sRGB, que fue el que mejor resultados obtuvo. Descarté utilizar los perfiles de color en escala de grises porque, al tener únicamente una componente, la impresora interpreta que no debe utilizar negro compuesto e imprime en negro puro. Los resultados han sido los siguientes:

- **Perceptual:** Es el valor por defecto, así que el resultado es el mismo que en las pruebas de perfil de color.
- **Saturation:** El negro sale muy brillante, pero en los grises intermedios se pueden apreciar tonalidades de color no deseadas.

Consideraciones: Es totalmente desaconsejable utilizar espacios de color LAB o XYZ, debido a que no son espacios de color adecuados para impresoras y los resultados no son buenos. Además, utilizar métodos de renderizado que modifiquen los niveles de gris, bien saturándolos o bien adaptándolos a la interpretación que les da nuestra vista, pueden producir cambios no deseados a la hora de imprimir las imágenes.

Los resultados obtenidos, aunque mejoran el estado anterior, no son del todo satisfactorios, ya que se siguen notando, aunque más sutiles, tonalidades rosadas en ciertas zonas de la imagen y verdes en otras. Debido a la mejora, el diagrama del flujo del sistema pasa a ser el descrito en la figura5.17.

Nueva generación del pdf

Tras realizar todos estos cambios, decido probar a crear directamente un pdf colocando las imágenes de igual forma que coloco los objetos del pdf plantilla. A pesar de la complejidad de crear un pdf e insertar directamente las imágenes, considero que esta aproximación puede ser muy interesante, debido a que cada librería tendrá una forma distinta de crear el documento y de insertar las imágenes. En la figura 5.18 se muestran las partes afectadas del sistema en esta sección.

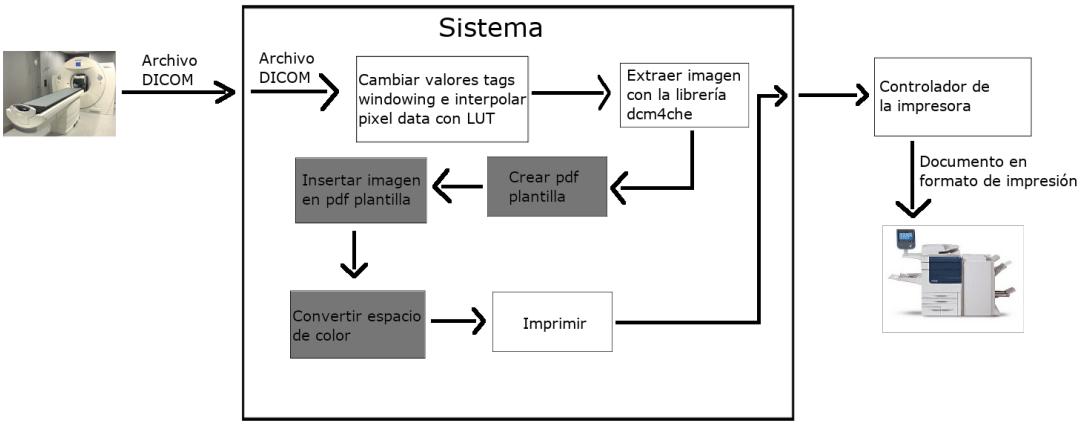


Figura 5.18: Diagrama de la estructura del programa

Para poder comparar correctamente los resultados de ambas generaciones de pdf, he tenido que suprimir el paso de convertir el espacio de color del pdf, ya que este paso solo lo podía realizar teniendo abierto el pdf con la librería AdobePDFLibrary.

Consideraciones: Se aprecian ciertas diferencias a la hora de imprimir las imágenes: las impresas con la generación de la librería de adobe presentan una tonalidad más clara y rosada, mientras que las impresas con la generación de PdfBox mantienen la calidad de color que se había conseguido hasta antes del paso de conversión del espacio de color (ver punto 5.1.1). En cuanto a resolución y apreciación de detalles no se aprecian diferencias. Cabe destacar que aplicando un perfil de color al pdf generado con la librería de Adobe, se obtiene un resultado muy parecido al generado con PdfBox; simplemente se aprecia una tonalidad un poco más clara en los colores oscuros.

Las figuras 5.19 y 5.20 representan la estructura del sistema al utilizar la librería AdobePDFLirary y al utilizar la librería PdfBox respectivamente.

Debido a que el resultado de imprimir un pdf generado con PdfBox sin pasar por una plantilla es igual al de generar una plantilla intermedia, decido suprimir el paso de generación de plantilla, ya que simplemente aumenta el tiempo de ejecución.

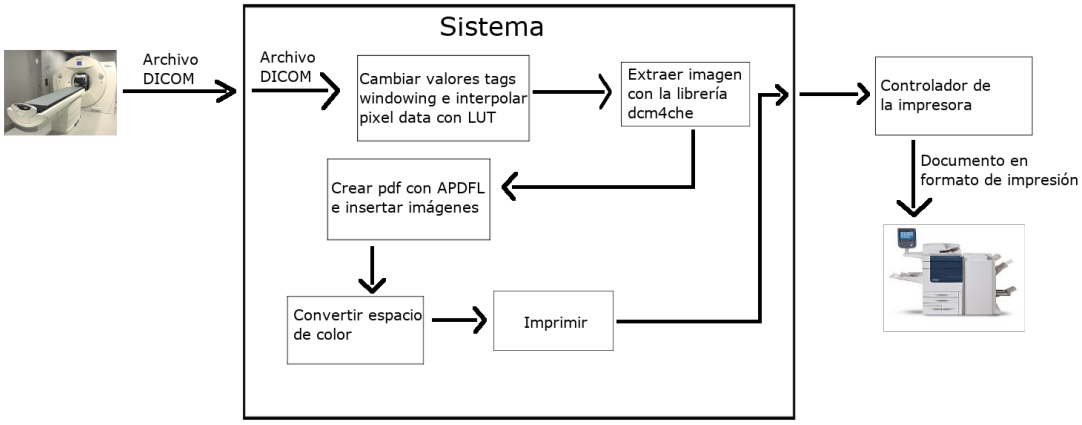


Figura 5.19: Diagrama de la estructura del programa usando AdobePDFLibrary

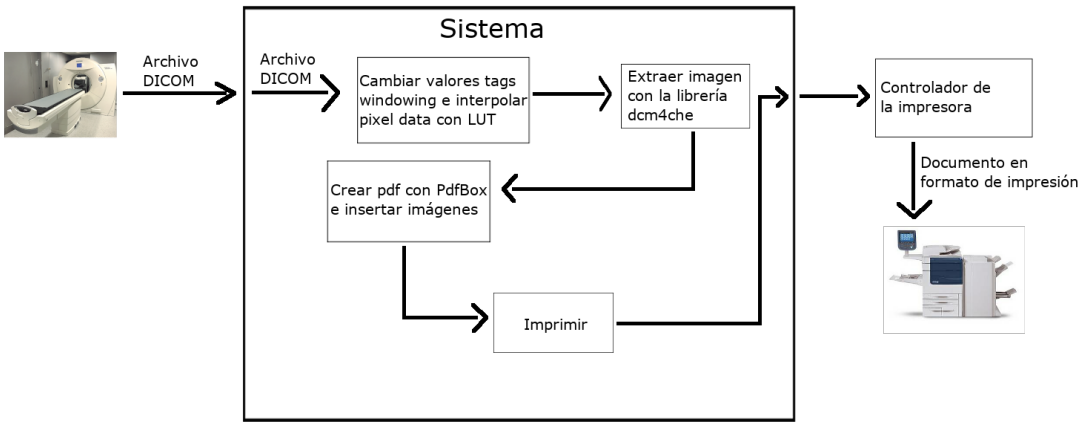


Figura 5.20: Diagrama de la estructura del programa usando PdfBox

Nuevos formatos de impresión (dejando pdf a un lado)

Tras probar todas las opciones que me ofrece esta librería, decido investigar más formas de mejorar el sistema para evitar perder información. Para ello, comienzo por suprimir un paso que puede acarrear una pérdida de información: En el sistema actual, tras generar el pdf, éste se guarda en un historial de documentos generados. El sistema, en lugar de continuar la ejecución con el documento ya en memoria, lo que hace es cargar de nuevo el pdf del archivo en el historial. Este paso, aunque parece innecesario, es útil para aumentar la modularidad del sistema; sin embargo, en nuestro caso queremos mantener la máxima calidad de imagen, por lo que no podemos arriesgarnos a que no sufra modificaciones durante el proceso de guardado y cargado del pdf. Tras realizar una modificación en el sistema para utilizar el documento en memoria en lugar del ya guardado en disco, se imprime una imagen de prueba pero no se aprecia ninguna diferencia a simple vista.

Las partes del sistema que se tratarán en este apartado se ven resaltadas en la figura 5.21. Nótese que he resumido los distintos pasos de generación del pdf representados en las figuras 5.19 y 5.20 en uno solo denominado ”generación pdf”.

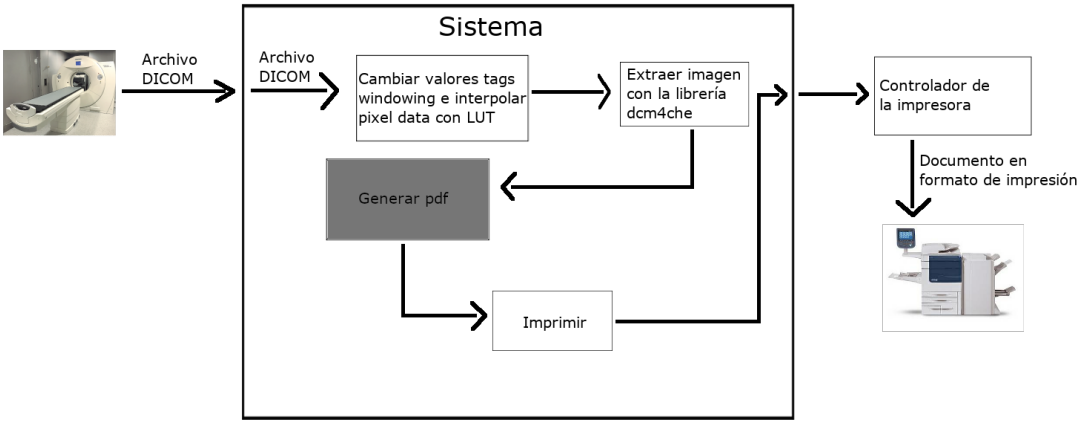


Figura 5.21: Diagrama de la estructura del programa

Continuando con la investigación, decido probar distintos formatos de impresión para no depender del formato pdf por si la culpa de la disminución de calidad y cambio de color era suya. Comencé buscando distintos formatos de impresión de imágenes. Cabe destacar que para generar un book (ver punto 2.1.2), las imágenes han de ser impresas en el mismo trabajo de impresión, por lo que no sirve, por ejemplo, imprimir individualmente varios archivos png.

El formato de imagen más indicado para realizar este trabajo es el formato tiff. Tiff es un formato de imagen que, por defecto, no realiza ninguna compresión en la imagen y permite archivos con varias páginas de imágenes. Por desgracia, tras configurar el sistema para generar archivos tiff con la imagen extraída del pixel data, el resultado no es tan satisfactorio como el ya conseguido con pdf, ya que se aprecian ciertas zonas con exceso de magenta y de amarillo.

Buscando más optativas al pdf, encuentro que la librería de Java con la que se imprime el trabajo (java.awt.print), tiene una interfaz Printable. Dicha interfaz permite crear una clase con un método en el que se define cómo se ha de imprimir un objeto de dicha clase. Para probarlo, creé la clase BookPrintable, que implementa la interfaz Printable y, por tanto, su método print(); en dicho método se establece la forma en que ha de imprimirse cada página del documento. Por ello, antes de imprimir el objeto guardo en él las imágenes del estudio, junto con su posición en la página y su número de página. Después, al imprimirlo, automáticamente el objeto sitúa las imágenes en su posición sin ningún tipo de escalado. Este método de impresión tiene una gran ventaja, y es que en todo momento puedes controlar lo que se imprime y cómo se imprime; sin embargo, su principal desventaja es la necesidad de implementarlo y su complejidad.

Consideraciones: Los resultados obtenidos usando la clase BookPrintable fueron los mejores hasta el momento, tanto en calidad de color como en calidad de detalles, supongo que por la compresión que realiza el formato pdf. Dicha compresión es fácilmente apreciable a la hora de enviar el trabajo a la impresora, ya que mientras un archivo pdf que ocupaba 20mb, el mismo trabajo enviado como un objeto BookPrintable ocupa 50mb.

Por lo tanto, en el entorno en el que he realizado el análisis, puedo concluir que el mejor método para imprimir las imágenes es utilizar la librería propia de Java y obviar la generación e impresión de un archiv pdf. Por consiguiente, la estructura del programa tras esta parte del análisis es la que se muestra en la figura 5.22. Como se puede apreciar, el paso de convertir el espacio de color se omite porque era aplicable solamente sobre un documento pdf.

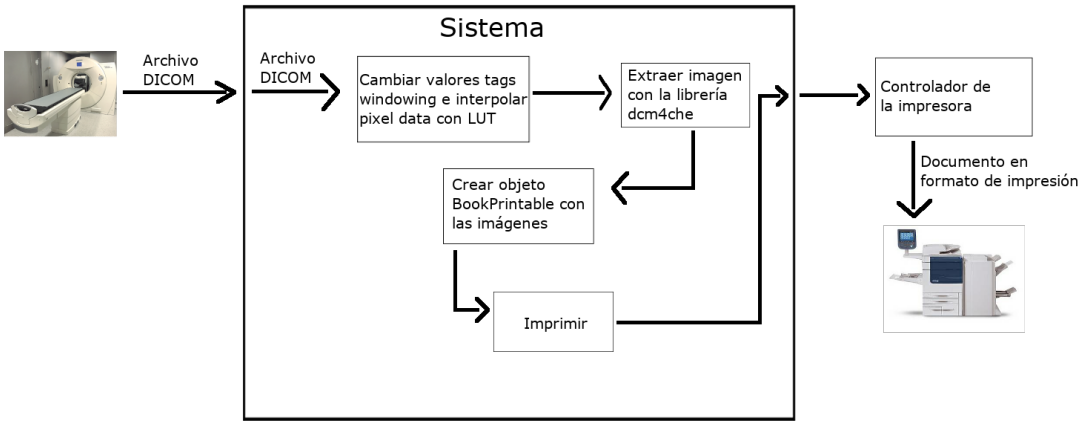


Figura 5.22: Diagrama de la estructura del programa

Perfiles de color sobre las imágenes

Esta forma de imprimir un trabajo abre nuevas posibilidades como, por ejemplo, cambiar el espacio de color de las imágenes, ya que hasta ahora era imposible cambiar el espacio de color de una imagen antes de incrustarla en el pdf, debido a problemas de compatibilidad al intentar incrustar la imagen con cierto espacio de color en un pdf con otro espacio. La única forma era utilizando la librería AdobePDFLibrary, pero además de aplicar la transformación sobre el pdf en lugar de sobre las imágenes, los resultados no fueron satisfactorios. Además, la librería tiene una cantidad limitada de perfiles de color. Gracias a la clase BookPrintable, podemos modificar la imagen para utilizar cualquier perfil de color ICC [50], e incluso podríamos usar un perfil de color distinto para cada imagen.

En este momento comencé a buscar perfiles de color, y encontré algunos perfiles CMYK con una documentación bastante completa de sus propiedades en la página oficial de International Color Consortium (ICC) color.org. Entre dichas propiedades se encuentran el tipo de

papel para el que están hechos y el porcentaje de cada color que se aplica para crear el negro compuesto, por lo que es importante contar con una documentación al respecto.

Los perfiles de color CMYK probados han sido: CoatedFOGRA27, CoatedFOGRA39, CoatedGRACoL2006, JapanColor2001Coated, JapanColor2001Uncoated, JapanColor2002Newspaper, JapanColor2003WebCoated, JapanWebCoated, UncoatedFOGRA29, USWebCoatedSWOP, USWebUncoated, WebCoatedFOGRA28, WebCoatedSWOP2006Grade3 y WebCoatedSWOP2006Grade5.

Los dos perfiles de color que mejores resultados mostraron han sido JapanColor2001Coated y JapanWebCoated, ya que fueron los que consiguieron casi compensar por completo las tonalidades de color producidas hasta el momento.

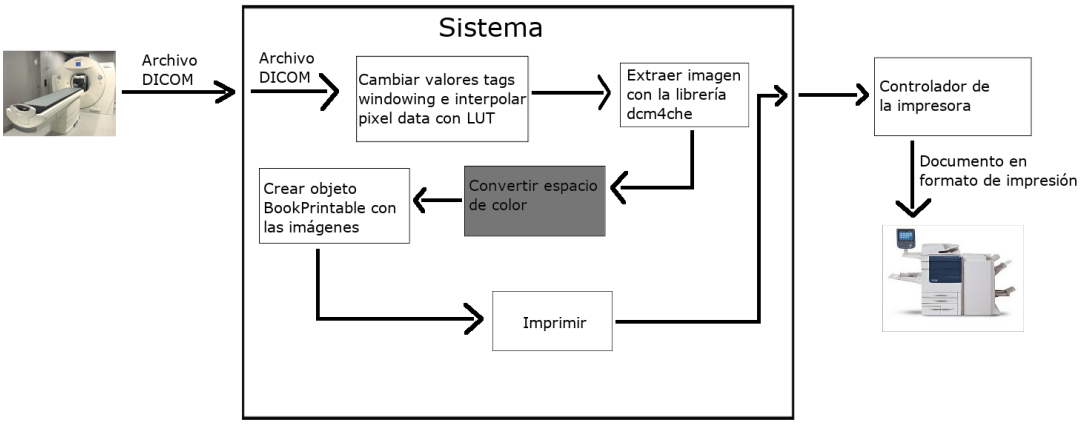


Figura 5.23: Diagrama de la estructura del programa

En la figura 5.23 se muestra la nueva estructura del programa tras la aplicación de los perfiles de color sobre las imágenes y se resalta la nueva parte de la estructura. que es la parte donde se han realizado las pruebas de esta sección.

Consideraciones: Como ya se ha comprobado, aplicar un perfil de color permite mejorar la combinación de colores que realiza la impresora para tratar de crear un negro compuesto. Sin embargo, dicho perfil no siempre es el mismo, ya que dependiendo del papel en el que se imprime, la impresora utilizada y la luz del entorno en el que nos encontremos, la apreciación de los colores será diferente.

Extracción manual del pixel data

En esta sección se trata la extracción de la imagen del archivo DICOM. En la figura 5.24 se muestran resaltados los pasos afectados.

Tras comprobar que cambiando el perfil de color de la imagen se mejoran los resultados, decido ir un paso más allá y tratar de extraer la imagen del archivo DICOM directamente en

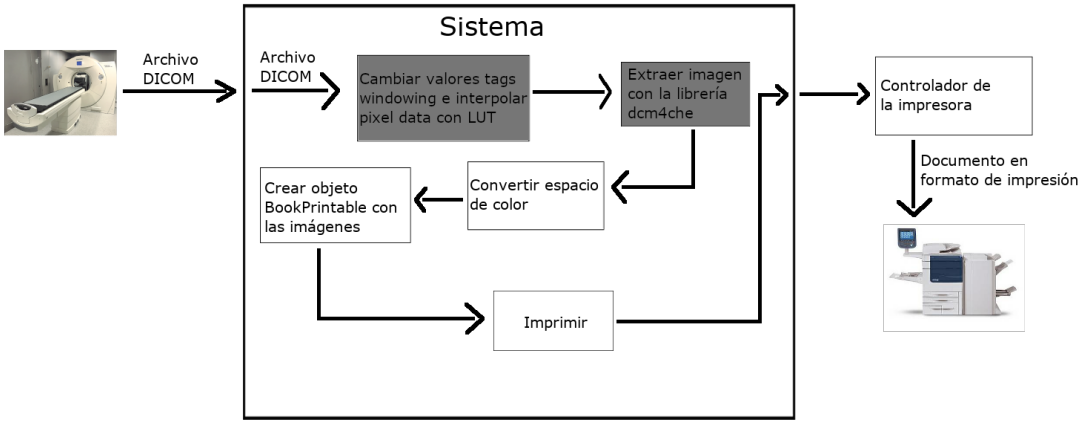


Figura 5.24: Diagrama de la estructura del programa

un espacio de color CMYK, ya que las herramientas proporcionadas por la librería dcm4che la extraen directamente como un objeto BufferedImage y con un espacio de color RGB, lo cual puede provocar la pérdida o transformación indeseada de ciertos colores al tratar de convertirlo más tarde a un espacio de color CMYK. Por desgracia, tras probar todas las herramientas proporcionadas por la librería dcm4che, ninguna me permite extraer la imagen en un determinado espacio de color. Finalmente, decido extraer el pixel data como si se tratase de cualquier otro tag, ya que de esta forma la librería me permite extraerlo como una simple matriz de números, y realizo manualmente la interpolación con el windowing y la tabla LUT de valores de pixel data a valores de gris. Esto me permite tener un mayor control sobre la extracción de la imagen y abre muchas nuevas posibilidades. Entre ellas, aplicar una corrección manual del color a cada componente, elegir el número de componentes de color que debe tener la imagen, aplicar directamente un perfil de color propio, etc.

Tratando de compensar manualmente los colores para evitar las tonalidades de color en la imagen impresa, me doy cuenta de que la complejidad es muy grande, ya que para cada nivel de gris la mezcla de colores es diferente y por tanto la tonalidad del color es distinta. Analizar todas las tonalidades de gris y poder sacar la corrección de color para una de ellas conlleva una cantidad de tiempo enorme, por no mencionar que varía dependiendo de cada equipo o incluso del nivel de tóner de la impresora, por lo que determino que no vale la pena.

Una idea que parece prometedora pinta es crear un perfil de color para cada impresora y cada ordenador, o incluso para cada modalidad. Por desgracia, buscando información sobre ello me doy cuenta de que crear un perfil de color correctamente está fuera del alcance de mis conocimientos, y no tengo el tiempo necesario para adquirirlos, por lo que descarto la posibilidad y lo dejo para un posible trabajo futuro (ver punto 7.5).

Consideraciones: A pesar de que en la imagen impresa no se aprecia ninguna diferencia con respecto a la imagen extraída directamente con la librería, interpolar el pixel data manualmente abre una nueva ventana de posibilidades y no parece afectar al rendimiento, por lo que tras consultarlo con el responsable del sistema existente, decidimos dejar la interpolación manual por si es necesaria alguna modificación en un futuro. Además, la interpolación manual permite aplicar la LUT como es debido para interpolar el pixel data a valores de 0 a 255, no a los mismos valores como se hacía hasta ahora.

Por lo tanto, tras estos cambios, la estructura del programa es la reflejada en la figura 5.25.

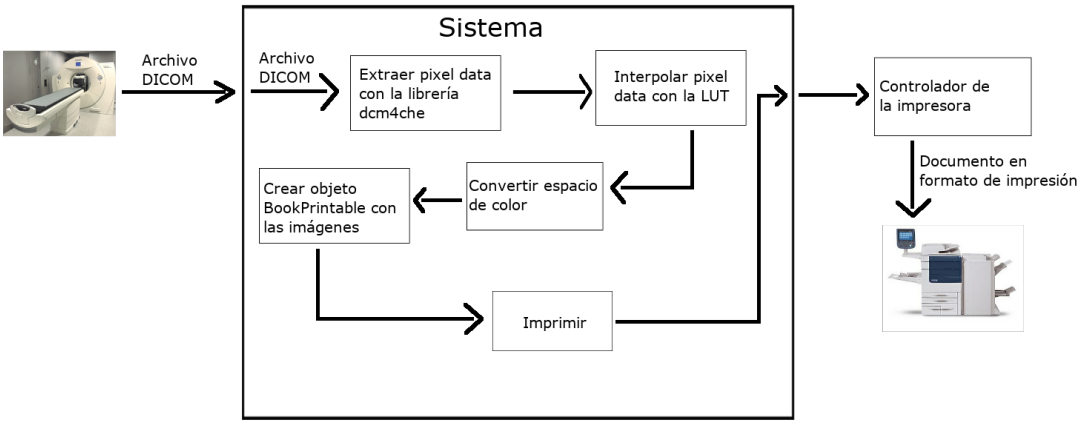


Figura 5.25: Diagrama de la estructura del programa

Juntando PDF/X y los perfiles de color

Tras adquirir toda la información sobre los perfiles de color y las interpretaciones de los pdf, decidí volver a darle una oportunidad al formato PDF/X, juntándolo con cambios en el perfil de color del documento. Para ello, tengo que hacer uso de la librería AdobePDFLibrary de nuevo y aplicar las opciones de conversión de espacio de color del documento (ver punto 5.1.1) sobre un documento PDF/X., por lo que el diagrama del sistema para realizar las pruebas es el de la figura 5.26. Como podemos observar, es el mismo que el de la figura figura 5.19 pero transformado en PDF/X.

El estándar PDF/X solo permite espacios de color CMYK, por lo que estoy restringido a dos perfiles de color: (Actobat_5_CMYK y Acrobat_9_CMYK). Probando ambos perfiles junto con las distintas opciones de renderizado, ninguno provee un resultado satisfactorio, ya que todas las pruebas salen oscuras y con falta de detalle, con lo cual descarto definitivamente el uso de PDF/X en este entorno.

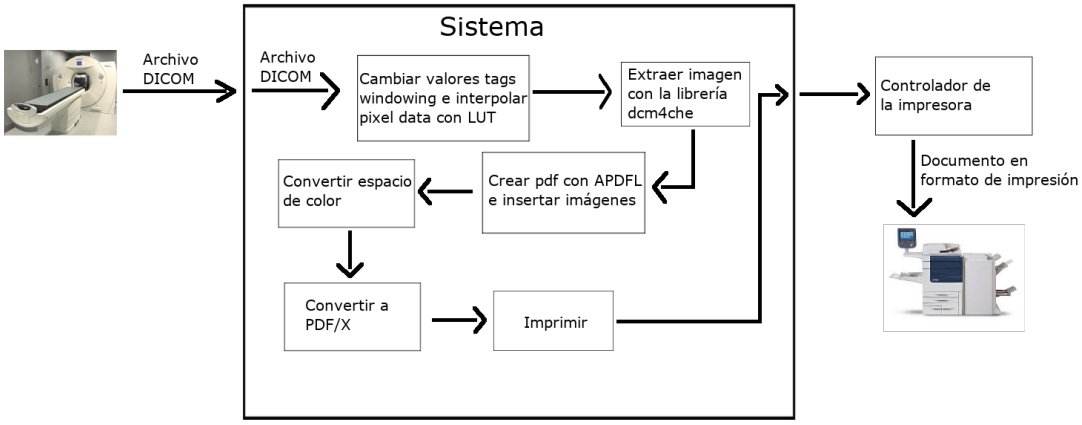


Figura 5.26: Diagrama de la estructura del programa

Cambios en el controlador de la impresora

El controlador ofrece muchas opciones de corrección del color, entre las que podemos encontrar correcciones del color específicas para gran parte de los perfiles de color ICC. Por desgracia, tras probar algunas de las correcciones de color y viendo sus nefastos resultados, decido dejar la corrección automática por defecto.

Además de las correcciones específicas para los perfiles de color, existen opciones de corrección manual. Dichas correcciones tienen un resultado muy parecido a la corrección manual de la interpolación del pixel data explicada en el punto 5.1.1, y el mismo gran inconveniente, y es que aparecen reflejos de distintos colores dependiendo del nivel de gris, por lo que resulta imposible ajustar una corrección manual.

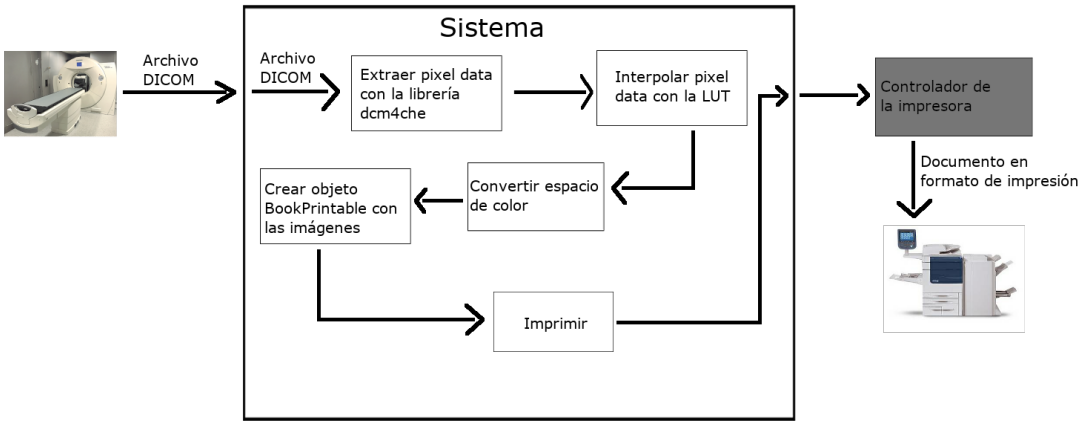


Figura 5.27: Diagrama de la estructura del programa

En la figura 5.27 se resalta la función del controlador en el proceso. Cabe destacar que el

proceso que se muestra es el que mejor resultados ha dado hasta el momento.

Guardando un objeto BookPrintable

Es una práctica habitual volver a imprimir un estudio ya impreso, por ello existe un histórico y es común acceder al historial y seleccionar un trabajo ya impreso para reimprimirlo. Por ello, para no perder la calidad que ofrece imprimir un objeto BookPrintable, debo desarrollar algún método que permita volcar toda la información del objeto en un archivo para guardar todas las imágenes que contiene y la forma en que han sido impresas.

Para ello, hago uso de la interfaz Serializable. Dicha interfaz proporciona, a todas las clases que la implementan, una sencilla manera de volcar todo el contenido del objeto perteneciente a dicha clase a un archivo, por conveniencia, con extensión .ser. Por lo tanto, creo la clase BookSerializable, que además de los métodos convenientes para guardar los datos y volcarlos, contiene también un método que permite transformar un objeto de la clase BookSerializable en BookPrintable. Creo en la clase BookPrintable un método homólogo para transformar un objeto en BookSeriailzable

Además, existe la necesidad de guardar un archivo pdf del trabajo, por lo que hago uso de la librería IText, que es la única que me permite crear un pdf a partir de un objeto Graphics, que es el objeto que se utiliza en la clase BookPrintable para definir la forma en que se imprime un objeto de la misma clase. Parece que la generación del pdf con IText no empeora notablemente la calidad, por lo que parece aceptable. Aprovechando que tengo que incorporar la librería IText, pruebo a imprimir documentos utilizando esta librería para cargarlos. No puedo apreciar cambios con respecto a PrinterJob.

Conclusiones del análisis

Es muy sencillo imprimir una imagen médica si es para un propósito general, pero a la hora de imprimirla para que un médico prescriptor pueda confirmar un diagnóstico, entran un juego multitud de factores que considerar. Para ello, es necesario crear un sistema que permita extraer la imagen de un archivo DICOM y procesarla de tal forma que se pierda la menor cantidad de información posible durante su impresión.

La primera fase es extraer la imagen del archivo DICOM. Esta fase es la clave para conseguir que en la imagen impresa se aprecien tantos detalles como se aprecian en el monitor y permitir al médico visualizar aquella parte de la imagen que debe visualizar. Para esta fase es fundamental establecer unos buenos valores de windowing y LUT, y para ello, lo más importante a tener en cuenta son la modalidad, el área del cuerpo que se examina y el diagnóstico.

La segunda fase es manipular la imagen extraída. Esta fase es fundamental para mantener la calidad de imagen proporcionada por la modalidad. A la hora de manipular la imagen es esencial evitar cualquier tipo de escalado, ya que provoca modificaciones no deseadas sobre la

imagen. Además, es conveniente no cambiar de soporte dicha imagen, por ejemplo, guardándola como imagen .jpg o .png, ya que los cambios de soporte suelen conllevar compresiones en la imagen, y dichas compresiones conllevar reducciones de calidad. En el caso de ser necesario guardar la imagen médica como un archivo de imagen, se recomienda utilizar el formato .tiff, ya que no utiliza ningún algoritmo de compresión. En el caso de requerir comprimir los archivos por necesidad de espacio en disco, se recomienda utilizar .png.

En el caso de que las imágenes se impriman con una tonalidad no deseada, se recomienda probar diferentes perfiles de color durante la manipulación de la imagen, teniendo en cuenta las propiedades de cada perfil a la hora de hacer la mezcla de colores para crear grises y la tonalidad de color de la imagen impresa. En nuestro sistema, los perfiles de color con mejores resultados han sido JapanColor2001Coated y JapanWebCoated.

Lo siguiente es generar un documento imprimible. En esta fase hay que tener un especial cuidado, ya que está en juego perder la calidad que se ha mantenido hasta este punto. El documento ha de ser creado con las dimensiones adecuadas, de no ser así, el controlador de la impresora o la impresora podrían escalar el documento, lo cual puede conllevar una pérdida de calidad o una mala interpretación de las proporciones de la imagen. Además, a la hora de insertar las imágenes hay que asegurarse de que no se escalen y se inserten en un rectángulo del documento con las mismas proporciones, ya que también podrían conllevar una pérdida de calidad o una mala interpretación de las proporciones.

La generación del documento también puede generar artefactos o malas interpretaciones del color por parte de la impresora, por lo que es muy importante hacer pruebas de generación para averiguar cuál es la que mejor resultados da en ese entorno.

Las principales opciones para crear el documento son:

- Crear un documento PDF/A, el cual será compatible con prácticamente cualquier impresora y ofrecerá una buena calidad de impresión. Además, es la opción más sencilla de implementar.

- Crear un documento PDF/X. Esta opción es recomendable si la impresora es compatible con PDF/X y el documento PDF/A se imprime con tonalidades de color no deseadas. Es sencillo de implementar pero por lo general produce artefactos en la imagen y es imprescindible calibrar la imagen estableciendo buenos valores de windowing y LUT para que se imprima lo mejor posible.

- Crear un documento propio, definiendo la forma en que se imprimirá. En nuestro caso, debido a que el sistema está implementado en Java, haciendo uso de la clase Printable de la librería java.awt.print. Dicha clase nos permite definir la forma en que se imprimirá el documento creado. Esta opción es la recomendable, debido a que ofrece mayor libertad para crear el documento y evita la compresión que pueda realizar el formato PDF. Esta

opción es más difícil de implementar que las anteriores, pero ofrece mejores resultados y más posibilidades de adaptación al entorno.

El último paso, y el más importante, es imprimir el documento. A la hora de imprimir el documento entran en juego tres elementos: los parámetros de impresión del sistema con el que se imprime, el controlador de la impresora y la impresora.

Los parámetros de impresión dependen del sistema con el que se imprime el documento, y sus resultados varían dependiendo del entorno y del documento, por lo que no existe un parámetro secreto que nos permita imprimir bien en cualquier entorno. En nuestro caso, lo recomendable es dejar estos parámetros por defecto a no ser que algún parámetro permita directamente aumentar la calidad de las imágenes (por ejemplo elegir los puntos por pulgada).

En nuestro caso, las opciones para imprimir el documento han sido utilizar las librería java.awt.print y AdobePDFLibrary. Debido a que la librería AdobePDFLibrary es de pago, difícil de utilizar y los resultados obtenidos con ella no son suficientemente satisfactorios, desaconsejo utilizarla a no ser que los resultados en el entorno específico sean claramente mejores.

En cuanto al controlador de la impresora, si alguna opción permite elegir la calidad o resolución de la imagen, se debe seleccionar la mejor resolución disponible. Además, si el controlador te permite elegir entre usar negro puro o negro compuesto, se recomienda encarecidamente utilizar negro compuesto, ya que ofrece un resultado mucho más realista y permite una mejor diferenciación entre valores de gris adyacentes. La opción de utilizar el negro puro se recomienda únicamente cuando el resultado tenga una cantidad de color que impida interpretar el diagnóstico.

La impresora utilizada determinará la calidad final de impresión. Cuanto más alta sea la gama de la impresora, mejor será la calidad de impresión. Se recomienda utilizar impresoras láser, debido a que las de gama alta ofrecen una calidad profesional (2400 ppp), un costo por impresión bajo y una duración elevada.

5.1.2 Diseño

Todas las pruebas realizadas en la fase de análisis se han diseñado sobre el sistema y modificándolo, por lo que es funcional. Sin embargo, para garantizar su correcto funcionamiento se propone un nuevo diseño que implemente los cambios. Cabe destacar que para el desarrollo de esta aplicación ha sido necesario modificar por completo todas las clases del sistema, así como incluir nuevas. Además, tanto una correcta estructuración como la existencia de diagramas permitirán a un trabajo futuro comprender mejor el sistema para realizar modificaciones o mejoras.

Diagrama de Clases

En la figura 5.28 se puede ver un diagrama con las clases del sistema y sus relaciones. En el anexo C se puede ver el diagrama de clases con atributos y métodos.

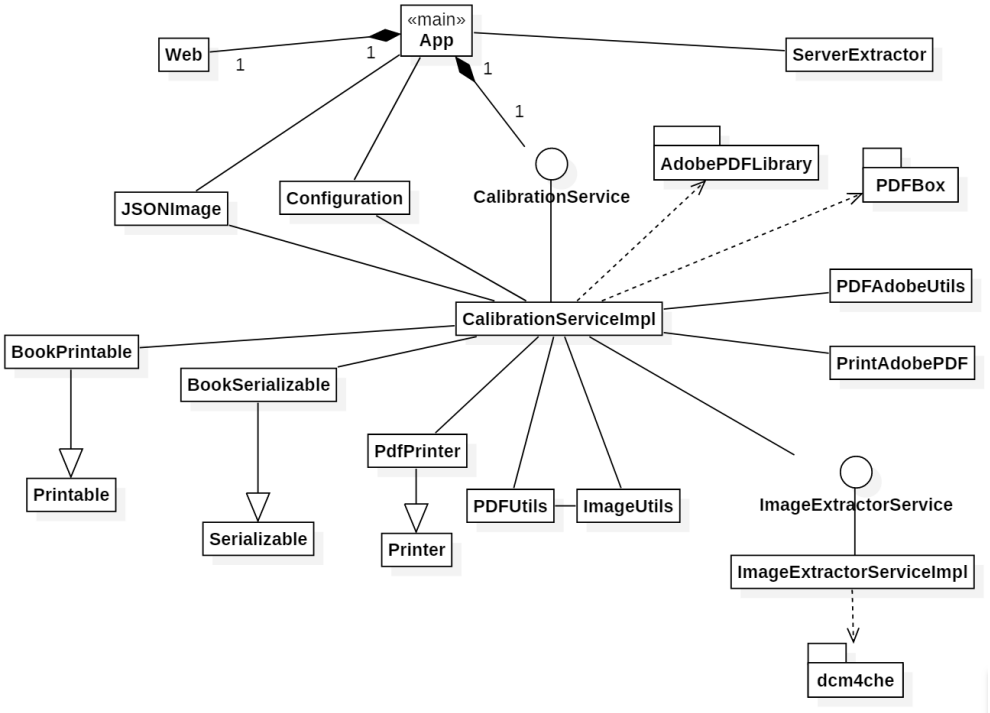


Figura 5.28: Relaciones entre las clases de la primera iteración

- **Web:** Representa el sistema de calibración, desarrollado en la segunda iteración (ver punto 5.2).

- **App:** Es la clase principal o 'main' del sistema. A su vez, actúa de controlador del patrón model-view-controler (explicado en el punto 5.3.1). Esta clase hace de intermediario entre el PACS o la herramienta de calibración y el resto del sistema. Es decir, atiende las peticiones HTTP y actúa en consecuencia, típicamente redirigiéndolas al modelo (clase CalibrationService).

- **CalibrationService:** Modelo del patrón model-view-controler (explicado en el punto 5.3.1). Es una interfaz que determina todos lo métodos que debe implementar cualquier clase que la implemente. Contiene un métodos para impresión, generación y guardado de Books, así como un método para extraer las propiedades de la imagen de un archivo DICOM como un objeto JSONImage.

- **CalibrationServiceImpl:** Única implementación de la interfaz CalibrationService, donde se desarrollan cada uno de sus métodos. Se comunica con ImageExtractorService para extraer cualquier tag o atributo de un archivo DICOM.
- **ImageExtractorService:** Interfaz que determina todos lo métodos que debe implementar cualquier clase que la implemente. Contiene un método para extraer una imagen de un archivo DICOM con una LUT dada, otro para extraer el Pixel Data de un archivo DICOM y otro para extraer el Windowing de un archivo DICOM.
- **ImageExtractorServiceImpl:** Única implementación de la interfaz ImageExtractorService, donde se desarrollan cada uno de sus métodos. Esta es la única clase que interactúa directamente con archivos DICOM, y por lo tanto es la única que utiliza la librería dcm4che. En esta clase se realizan las interpolaciones para los valores de Pixel Data. Es decir, esta es la clase donde cualquier imagen DICOM pasa a tener valores de 0 a 4000 para su calibración en el navegador y más tarde se le aplica la LUT dada por la herramienta de calibración para que pase a tener valores de 0 a 255 y pueda ser impresa.
- **ServerExtractor:** Clase de utilidades usada por la clase App para ”traducir” los objetos JSON que se reciben con peticiones HTTP en objetos que ”entienda” el modelo, en este caso la clase CalibrationServiceImpl.
- **Configuration:** Clase que sirve de interfaz entre el sistema y el archivo de configuración config.xml. Cabe destacar que se ha implementado un observer (explicado en el punto 4.5.2) para poder modificar el archivo config.xml sin necesidad de reinicar el sistema.
- **JSONImage:** Clase que contiene todos los atributos necesarios para pasar a la herramienta de calibración como imagen DICOM. Sus atributos son todos compatibles con el formato JSON, en este caso usado para transferir datos entre Spring y JavaScript.
- **BookPrintable:** Clase que implementa la interfaz Printable del paquete java.awt.print. Dicha interfaz permite crear una clase que defina la forma en la que ha de ser impresa. En esta clase se almacenan todas las imágenes que van a ser impresas, junto con el rectángulo de la hoja que van a ocupar y la página en la que han de ser impresas. Cabe destacar que contiene un método estático que permite calcular el rectángulo que ocupará cada imagen en función de su posición (número de imagen dentro del estudio).

  Además, este objeto contiene un método que permite crear un objeto BookSerializable a partir del BookPrintable.
- **BookSerializable:** Clase que implementa la interfaz Serializable del paquete java.io. Dicha interfaz permite crear una clase que defina la forma en que ha de ser serializada,

lo cual en nuestro proyecto nos permite guardarla en un archivo con extensión .ser. Esto es útil para guardar un Book sin tener que recurrir a usar un pdf, transformación tras la cual no hay vuelta atrás.

Además, este objeto contiene un método que permite crear un objeto BookPrintable a partir del BookSerializable.

- **Printer:** Esta clase ya existía en el sistema previo y no ha sido modificada. En este sistema simplemente se usa para servir de superclase a PdfPrinter, ya que provee un constructor que busca la impresora adecuada o deseada.
- **PdfPrinter:** Esta clase ya existía en el sistema previo y sus métodos principales (printJob y printWithAdobeReader) apenas han sido modificados. Sin embargo, se han añadido dos nuevos métodos: printJavaDoc y printIText
- **PDFUtils:** Clase de utilidad que provee métodos para modificar un documento pdf, como por ejemplo añadir imágenes, texto, portadas, etc.

  Se excluyen los documentos de la librería AdobePDFLibrary.
- **ImageUtils:** Clase de utilidad que provee métodos para tratar imágenes.
- **PDFAdobeUtils:** Clase de utilidad que provee métodos para modificar un documento pdf como PDFUtils, pero aplicable solo a documentos pertenecientes a la librería AdobePDFLibrary.
- **PrintAdobePDF:** Clase que permite imprimir un documento pdf desde la librería AdobePDFLibrary.

Diagrama de Secuencia

Todos los diagramas de secuencia del resto del sistema se encuentran en el anexo F, punto F.2.

5.1.3 Codificación

Esta iteración ha sido codificada en Java, haciendo uso de Spring para comunicarse con el sistema de la segunda iteración. Para más información, ver el uso del patrón Model-View-Controller (punto 5.3.1).

5.1.4 Pruebas

Debido a la naturaleza y del sistema de esta iteración y a que durante el análisis ya se probaba el correcto funcionamiento, solo se han realizado pruebas a aquellos métodos lógicos

cuyo impacto pueda afectar al sistema, como por ejemplo la generación del rectángulo en el que se encajan las imágenes en el documento.

5.1.5 Conclusiones

Tras la implementación de esta primera iteración, podemos concluir que se ha mejorado la impresión con respecto a la versión anterior del sistema. Además, ahora se proveen métodos para que el usuario pueda elegir las herramientas a utilizar a la hora de generar e imprimir el Book, así como seleccionar distintos perfiles de color, todo ello en función de su entorno.

Por desgracia, aunque nos hemos acercado más, no se ha llegado a lograr el objetivo de visualizar igual la imagen en papel que en pantalla.

Además, algunos estudios no se imprimen correctamente y sigue siendo necesaria la herramienta de calibración explicada en el capítulo 3, y como ya vimos, dicha herramienta no es útil debido a su dificultad de uso, por ello, la segunda iteración se basará en crear una nueva herramienta de calibración que permita modificar el windowing y la LUT de una forma mucho más visual y sencilla.

5.2 Segunda Iteración

En el capítulo 3 se explica la utilidad de la herramienta de calibrado actual, así como sus defectos. Debemos recordar que dichos defectos dificultan tanto el uso de la herramienta que produce que los resultados no sean para nada satisfactorios, ya que es muy difícil calibrar la imagen como se desea. Por ello, la segunda iteración se centrará en el desarrollo de una nueva herramienta de calibración que supla los defectos de la herramienta anterior.

Durante el análisis se explicará más en profundidad la utilidad de la calibración y formas de realizarla.

5.2.1 Análisis

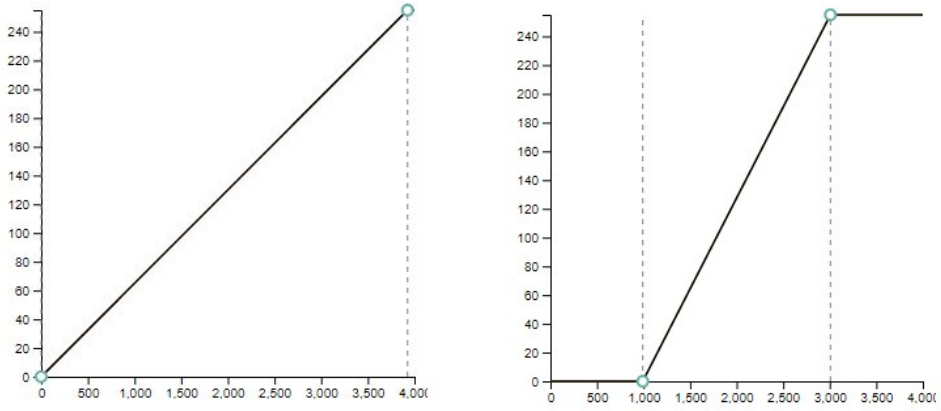
Para explicar el análisis necesario para crear la herramienta, comenzaré por profundizar más en la importancia del windowing y la LUT a la hora de calibrar e imprimir una imagen médica. Después hablaré de la generación de prototipos para crear la interfaz de usuario y extraer los casos de uso, que explicaré a continuación. Por último explicaré las distintas formas de modificar la función de interpolación o LUT que surgieron durante el análisis.

Importancia de windowing y LUT

Como ya se explicó anteriormente (ver punto 2.1.1), el windowing nos permite acotar el histograma de la imagen de forma que podamos ver mejor el rango de valores que hemos

acotado. Esto es debido a que los valores de pixel data que están fuera del rango marcado por el windowing no se interpolan a valores de gris, por lo que habrá más valores de gris ”disponibles” para los valores de pixel data dentro del rango del windowing.

En las figuras 5.29 se puede observar mejor cómo afecta el cambio del windowing a la interpolación de una imagen DICOM. En las imágenes podemos ver una interpolación lineal representada con una línea recta, y como se puede observar, los valores de la imagen, representados en el eje X con valores de 0 a 4000, se interpolarán a valores de 0 a 255, representados en el eje Y. Como se ve en la figura 5.29b, al disminuir el window width, la pendiente de la función aumenta en la zona restringida por el windowing, lo cual implica un aumento del contraste en esa zona, o lo que es lo mismo, un aumento del rango dinámico. Sin embargo, los valores fuera del windowing se interpolan como valores blanco o negro (0 o 255).



(a) Función de interpolación con windowing completo

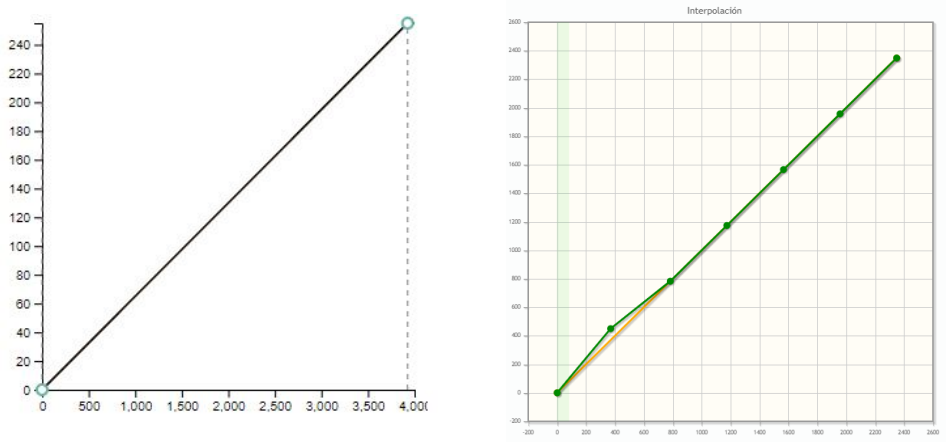
(b) Función de interpolación con decremento de window width

Figura 5.29: Cambio de la función de interpolación con respecto al windowing

Como ya he dicho anteriormente, los profesionales varían el windowing para poder ver la imagen en todas sus variables y poder detectar cualquier anomalía, pero como eso no se puede hacer en papel, necesitamos algún método que nos permita modificar solamente ciertos valores de gris para resaltar lo que se encuentra en ellos. Para ello entra en juego la LUT (ver punto 2.1.1.

Sin embargo, modificar una tabla entera es muy engorroso para cualquier posible usuario, debido a que en general nos encontramos con miles de valores de pixel data en cada imagen, por lo que haremos uso de una función que represente la tabla LUT y que llamaremos ”función LUT” (ver figura 5.30a). Dicha función define, para cada uno de los valores del pixel data (eje X), su valor en escala de gris (eje Y). Por lo tanto, para modificar la interpolación de ciertos valores de gris, basta con localizarlos en el eje X y modificar su correspondencia en el eje Y.

En la herramienta de calibración previa (ver capítulo 3) se extraía directamente el pixel



(a) Función LUT teórica

(b) Función LUT del sistema previo

Figura 5.30: Funciones LUT

data como una imagen con la librería dcm4che. Esto imposibilitaba realizar interpolación con la tabla LUT, ya que la librería realizaba una interpolación lineal por defecto. La solución aplicada en el sistema previo fue utilizar una variación de la tabla LUT para realizar una interpolación entre los mismos valores del pixel data. Como se puede ver en la figura 5.30b, el rango de valores en el eje X es el mismo que en el eje Y.

Gracias a los cambios en el proceso de extracción de la imagen médica de la primera iteración, ahora la LUT se puede aplicar como teóricamente corresponde, ya que se extrae el pixel data como una matriz de valores en vez de como una imagen, por lo que disponemos de los valores originales y podemos aplicar sobre ellos la LUT. En las figuras 5.31 se muestra gráficamente la diferencia de la interpolación para una mejor comprensión.

(a) Aplicación anterior de la LUT

(b) Aplicación actual de la LUT

Figura 5.31: Comparación entre la forma en que se aplicaba la LUT anteriormente y la actual

Además, en la herramienta previa se permitía cambiar el brillo y/o el contraste de la imagen general tras su extracción, lo cual puede producir una pérdida de información.

Podemos ver la pérdida de información en la función de interpolación. Como vemos en la figura 5.32, un aumento de brillo significaría "subir" en el eje Y todos los valores del eje X. Se puede observar que ahora el mínimo valor de gris tras la interpolación es casi 40, por lo que tenemos casi 40 valores de gris sin usar para interpretar una imagen de 4000 valores. Además, debido a que el valor máximo es 255, todos los valores que superan dicho número pasan a ser

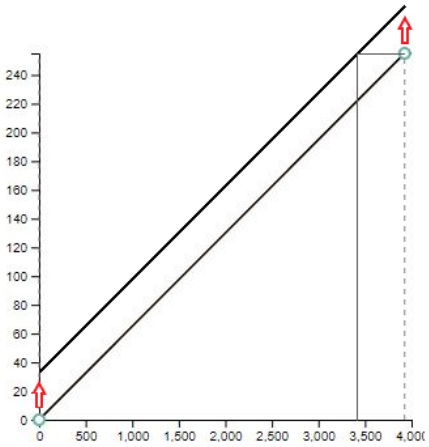


Figura 5.32: Pérdida de información al aumentar el brillo de toda la imagen

255, por lo que perdemos toda la información que se encuentra entre los valores 3400 a 4000 del pixel data.

Creación de prototipos

Para la creación del primer prototipo hay que tener varias cosas en mente:

- La herramienta ha de ser muy visual y la mejor forma de entender el windowing es haciendo uso del histograma
- La mejor forma que conocemos de representar la LUT es con una función
- La herramienta ha de ejecutarse en tiempo real para una buena experiencia de usuario

Afortunadamente, el primer prototipo fue suficiente para extraer todos los casos de uso de la herramienta (ver puntos 5.2.1 y 5.2.1).

Las imágenes del prototipo se encuentran en el apéndice A.

Gráfico de los casos de uso

El diagrama de casos de uso se muestra en la figura 5.33.

Como se puede observar en el diagrama, antes de realizar cualquier acción el usuario debe seleccionar un estudio para calibrar. Después, tendrá la posibilidad de modificar la función LUT, el Windowing, hacer zoom sobre la imagen del estudio, crear o restaurar un backup de calibración, o pedirle al sistema que imprima el Book o genere una vista previa.

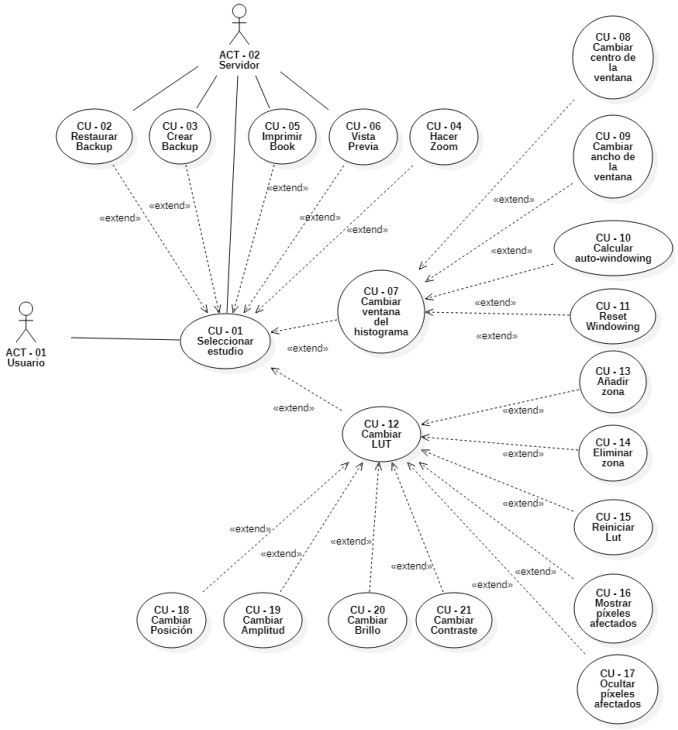


Figura 5.33: Diagrama de Casos de Uso

Definición de los casos de uso

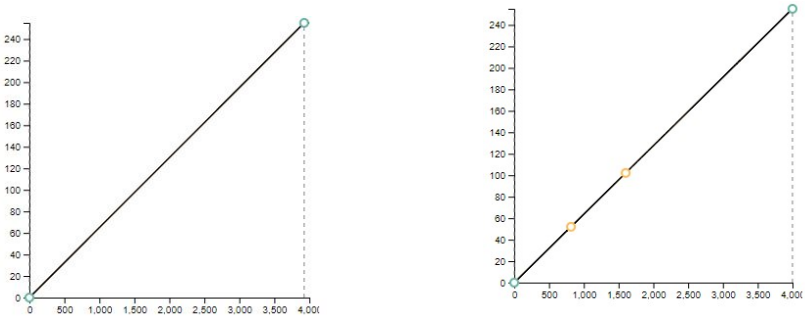
Los casos de uso están definidos en el anexo B.

Formas de modificar la función LUT

En la herramienta de calibración anterior, para modificar la función de interpolación se hacía uso de un gráfico en el que se puede mover cada punto en cualquier dirección (ver puntos 3.7 y 3.3). Esta es una mala forma de modificar la LUT, ya que el usuario no sabe exactamente lo que está haciendo y provoca que sea muy difícil calibrar una imagen.

Para ello, haremos uso de lo que llamamos ”Zonas LUT”, que no son más que segmentos de la función LUT que podremos modificar independientemente y que nos permitirán cambiar el brillo o el contraste de cierta zona del histograma de la imagen. El propósito general de utilizar las zonas LUT es disminuir los grados de libertad del usuario, obligándole a modificar la función bajo ciertas restricciones. En la figura 5.34a se muestra la función LUT por defecto, que como podemos ver, realiza una interpolación lineal. En la imagen 5.34b podemos ver la misma función pero con una Zona LUT, aunque en este momento dicha zona no modifica la función, representa el rango de píxeles del pixel data que podremos modificar.

Como se puede observar en los prototipos (ver punto 5.2.1), la única diferencia destacable



(a) Función LUT por defecto

(b) Función LUT con una Zona LUT

Figura 5.34: Cambio de la función de interpolación con respecto al windowing

con respecto al sistema final, se encuentra en el 'desplegable LUT', ya que en el prototipo se puede apreciar que existen solamente tres deslizables: posición, amplitud y magnitud. Esto es debido a que la idea inicial era mover cada 'Zona LUT' solamente en una línea perpendicular a la función LUT inicial, como se puede apreciar en la imagen 5.35.

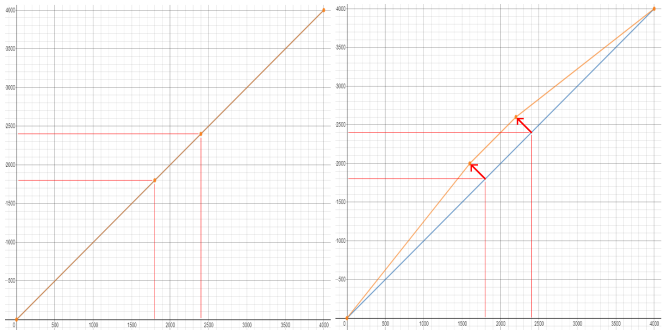


Figura 5.35: Ejemplo de modificación de la magnitud

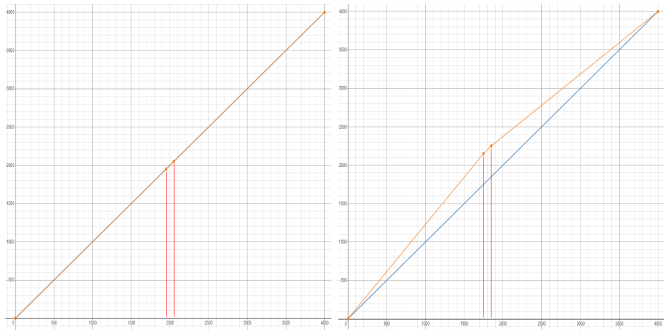
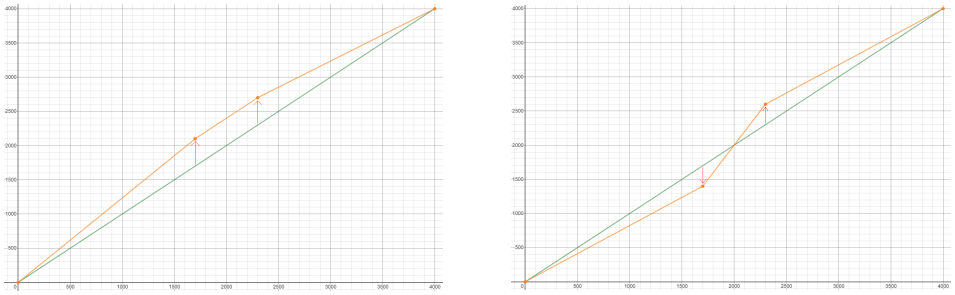


Figura 5.36: Ejemplo de modificación de la magnitud

Esta idea parecía buena al principio, sin embargo, me di cuenta de un gran problema que aumentaba en gran medida la complejidad tanto del problema como del programa, y es que

al aumentar la magnitud de una 'Zona LUT', el rango de píxeles a los que afecta dicha zona varía, como se puede apreciar mejor en la figura 5.36.

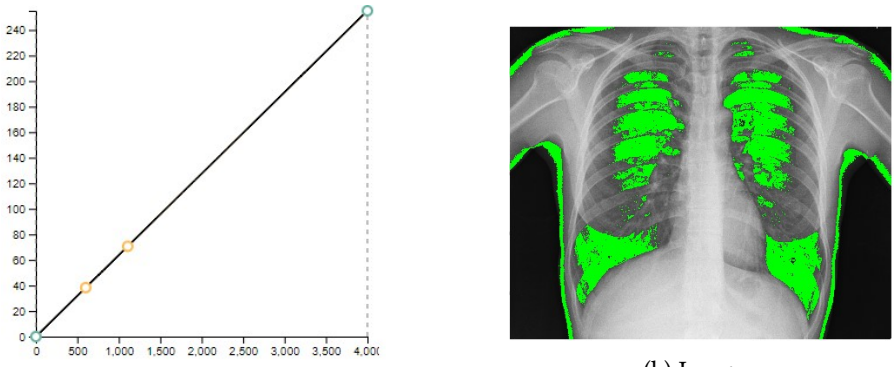
Por ello, finalmente decidí cambiar la forma en que se modifica una 'Zona LUT', optando por modificar los puntos de una forma más intuitiva para el usuario usando los conceptos de contraste y brillo. De esta forma, el usuario puede cambiar el contraste y el brillo de una zona en concreto de la función LUT sin siquiera conocer los conceptos de 'Zona LUT' o 'función LUT'. En la figura 5.37a se muestra un aumento de brillo, 'subiendo' ambos extremos de la 'Zona LUT', mientras que en la figura 5.37b se muestra un aumento de contraste, 'bajando' el extremo izquierdo de la 'Zona LUT' y 'subiendo' el derecho.



(a) Ejemplo de aumento de brillo (b) Ejemplo de aumento de contraste

Figura 5.37: Ejemplos función LUT

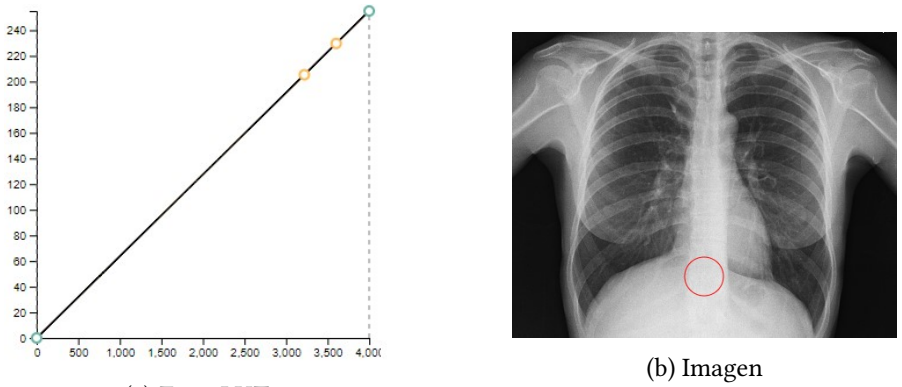
Para facilitar la comprensión de las zonas LUT, introduje un método que, bajo demanda del usuario, resalta los píxeles que están siendo afectados por la zona LUT actual. Ver figuras 5.38



(a) Zona LUT (b) Imagen

Figura 5.38: Ejemplo píxeles zona LUT

Además, para facilitar al usuario el uso de la herramienta, introduje una forma intuitiva de crear la zona LUT haciendo click en una zona de la imagen y arrastrando para crear un círculo. A partir de dicho círculo se crea una zona LUT en la que se encuentran la media de



(a) Zona LUT

(b) Imagen

Figura 5.39: Ejemplo círculo para crear Zona LUT

los píxeles seleccionados. Ver figuras 5.39.

5.2.2 Diseño

Tras el análisis y la extracción de requisitos, se procede con el diseño del sistema a implementar.

Diagrama de Clases

Como se puede observar en las figuras C.1 y 5.40, en el diagrama de relaciones hay algunas clases que no aparecen. Esto es debido a que dichas clases se usan como tipos de dato, es decir, son clases de utilidad, por lo que no he considerado relevante representar qué clase las usa.

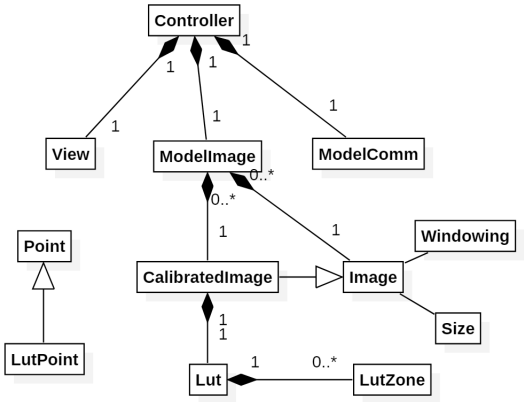


Figura 5.40: Relaciones entre las clases de la segunda iteración

- **Controller:** La clase Controller es la que actúa como controlador en el modelo model-view-controller (explicado en el punto 5.3.1). Su función es hacer de intermediario entre

la vista (clase View) y los modelos (clases ModelImage y ModelComm). Maneja cada una de las peticiones que se envían desde la vista y ejecuta el método del modelo correspondiente. Además, envía a la vista los datos que necesita para mostrárselos al usuario

- **View:** Esta clase es la que se encarga de recibir las peticiones del usuario y enviárselas al controlador, así como recibir los datos del controlador y actualizar los componentes HTML necesarios para mostrarle dichos datos al usuario.
- **ModelImage:** Esta clase se encarga de realizar todas las modificaciones necesarias sobre la imagen que se está calibrando en ese momento, así como almacenar toda la información necesaria para calibrar el estudio.
- **ModelComm:** La clase ModelComm es la que actúa como intermediario entre la herramienta de calibración y el resto del sistema. Es decir, esta clase envía a la aplicación que se ejecuta en el servidor todas las peticiones y recibe las respuestas. También almacena los datos necesarios para la comunicación con el servidor, como el id del estudio.
- **Image:** Esta clase representa la imagen DICOM. Contiene el Pixel Data por defecto (con valores entre 0 y 4000), el Windowing por defecto y las dimensiones de la imagen que se muestra en la interfaz del navegador.
- **CalibratedImage:** Esta clase, hija de Image, contiene además un atributo 'lut' de clase Lut, ya que representa la imagen calibrada, es decir, la que modifica el usuario. Contiene el Pixel Data calibrado (con valores entre 0 y 255), el Windowing que haya estipulado el usuario, las dimensiones y el ya mencionado atributo lut, que contiene toda la información de la función LUT que haya estipulado el usuario.
- **Lut:** En esta clase se almacena toda la información sobre la función LUT. Contiene una lista de las Zonas LUT y unos métodos útiles para representar la curva de la función LUT y añadir o eliminar zonas.
- **LutZone:** Esta clase almacena toda la información sobre una Zona LUT, como son su id o 'index', su posición, amplitud, brillo, contraste y el círculo que marca su posición.
- **Point:** Clase sencilla que representa un punto en dos dimensiones. Solo tiene dos atributos x e y.
- **LutPoint:** Esta clase, hija de Point, añade un nuevo atributo llamado color. Solo se utiliza para que la clase View sepa de qué color debe pintar cada punto de la función LUT.

En el anexo C se puede ver el diagrama de clases con sus atributos y métodos.

Diagrama de Flujo de Datos

Como se puede observar en el anexo D, han sido necesarios cuatro niveles incluyendo el nivel 0.

La descripción de cada uno de los procesos se explicará en el punto 5.2.2.

Especificación de Procesos

Las tablas de especificación de procesos se encuentran en el anexo E.

Diagrama de Secuencia

Todos los diagramas de secuencia se encuentran en el anexo F, punto F.1.

5.2.3 Codificación

La codificación se ha realizado usando JavaScript para la lógica del programa y HTML para definir la estructura de la interfaz. Cabe destacar que debido a que se ha mantenido el diseño para no desentonar con el resto del sistema, no se han producido modificaciones en los archivos CSS.

5.2.4 Pruebas

Las pruebas se han realizado con Jest sobre el sistema desarrollado en JavaScript.

5.2.5 Conclusiones

El resultado de esta iteración es satisfactorio, ya que cumple el objetivo principal de calibrar la imagen DICOM, así como los requisitos de simplicidad, facilidad de uso y rapidez.

Aunque la herramienta es mejorable, es un gran avance con respecto a la versión anterior y se dejan abiertas posibilidades para seguir mejorándola, como se explicará en el capítulo 7.

5.3 Patrones de Diseño Utilizados

Los patrones de diseño son una herramienta fundamental para el desarrollo software, ya que proveen soluciones de diseño a los problemas más comunes.

A continuación, explicaré los patrones que he tenido en cuenta a la hora de desarrollar el sistema, aunque no todos ellos hayan podido ser implementados.

5.3.1 Model-View-Controller

Aunque existen muchas variaciones del patrón MVC[51, 52], para este sistema he querido utilizar su versión más pura, por lo que no existe ningún tipo de comunicación entre el modelo y la vista. A pesar de ello, cabe destacar que como se explica en el punto 5.3.1, la segunda iteración hace uso de dos modelos.

MVC en la primera iteración

En la primera iteración, el rol de Vista lo desempeña la herramienta de calibración, ya que es la que se encarga de recoger las peticiones del usuario que deben ser ejecutadas en el servidor; y las recibe la clase App, que actúa como controlador.

El rol de Modelo lo desempeña la clase CalibrationService, y por tanto también su implementación CalibrationServiceImpl.

En la figura 5.41 se muestran gráficamente en un diagrama cada uno de los componentes.

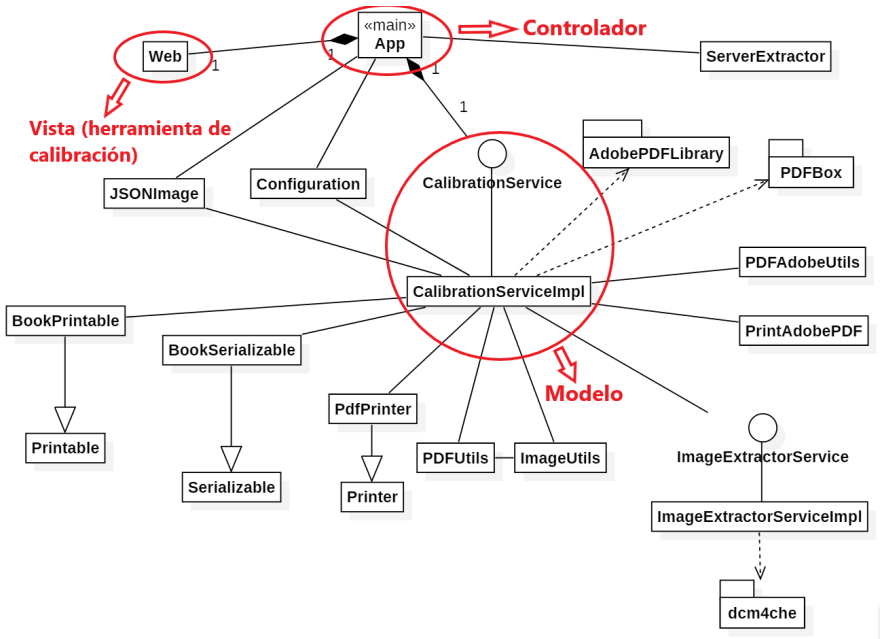


Figura 5.41: Model-View-Controller en la primera iteración

A continuación se describen las interacciones entre los componentes:

- El usuario interacciona con la vista, por ejemplo, pulsando un botón.
- El controlador recibe, por parte de a vista, dicha acción y gestiona el evento.
- El controlador transforma dicha acción en una petición al modelo.

- El modelo realiza las operaciones necesarias para satisfacer dicha petición y envía una respuesta al controlador de ser necesario.
- El controlador espera la respuesta del modelo, de ser necesaria, y envía a la vista la respuesta a la acción del usuario.
- La vista se actualiza conforme la respuesta del controlador.

MVC en la segunda iteración

Para implementar el patrón MVC en la segunda iteración, he decidido crear dos modelos, llamados ModelImage y ModelComm. Esta decisión se debe a la clara distinción de acciones que el modelo debe realizar, ya que por un lado debe ejecutar ciertos cambios en la interfaz en tiempo real y por otro debe enviar peticiones al resto del sistema para realizar todas aquellas acciones que se deben realizar en el servidor.

En la figura 5.42 se muestran gráficamente en un diagrama cada uno de los componentes.

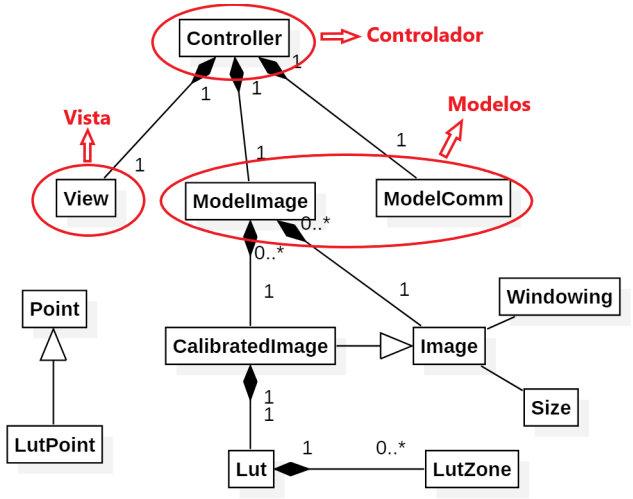


Figura 5.42: Model-View-Controller en la segunda iteración

Las interacciones entre los componentes siguen la siguiente forma:

- El usuario interacciona con la vista, por ejemplo, pulsando un botón.
- El controlador recibe, por parte de a vista, dicha acción y gestiona el evento.
- El controlador transforma dicha acción en una petición al modelo para transformar los datos acorde con la petición del usuario.

- El controlador pide los datos modificados al modelo (En el caso de ser ModelImage) o espera a que el modelo termine de realizar petición y espera por una respuesta (En el caso de ser ModelComm).
- El controlador envía un comando a la vista para que se actualice con los nuevos datos.

5.3.2 Patrón Observador

En el sistema, este patrón es utilizado en la primera iteración para mantener a las clases App y CalibrationServiceImpl al tanto de los cambios realizados en el archivo de configuración config.xml.

Para implementarlo, he utilizado las clases proporcionadas por la librería org.apache.commons.io.monitor:

- FileAlterationListener como Sujeto Concreto
- FileAlterationMonitor como Sujeto
- FileAlterationObserver como Observador

A continuación se muestra la parte del código en la que se implementa el patrón en la clase CalibrationServiceImpl:

```
1  // Creación del Observador
2  FileAlterationObserver observer = new
       FileAlterationObserver(CONFIG_FILE.getParentFile());
3
4  public CalibrationServiceImpl() {
5      // Creación del Sujeto Concreto
6    FileAlterationListener fal = new FileAlterationListenerAdaptor() {
7      @Override
8      public void onFileDelete(File file) {
9        System.err.println("Se ha borrado el archivo " +
       file.getName());
10     }
11
12     @Override
13     public void onFileCreate(File file) {
14       System.out.println("Se ha creado el archivo " +
       file.getName());
15       if (file.getName().equals(CONFIG_FILE.getName()))
16         config = Configuration.loadConfiguration(CONFIG_FILE);
17     }
18
19     @Override
```

```
20      public void onFileChange(File file) {
21        if (file.getName().equals(CONFIG_FILE.getName()))
22          config = Configuration.loadConfiguration(CONFIG_FILE);
23        System.out.println("Se ha modificado el archivo " +
        file.getName());
24      }
25    };
26
27      // Se asocia al Sujeto Concreto con el Observador
28    observer.addListener(fal);
29    // Creación del Sujeto
30    FileAlterationMonitor monitor = new FileAlterationMonitor(1); //
        comprueba cada segundo
31    // Se añade el observador al Sujeto
32    monitor.addObserver(observer);
33    try {
34      monitor.start();
35    } catch (Exception e) {
36      System.err.print("No ha sido posible iniciaizar el observer
        para " + CONFIG_FILE.getName());
37      System.err.print(". Reinicia el programa manualmente si
        realizas algún cambio");
38    }
39  }
```

5.3.3 Patrón Fachada

He decidido implementar el patrón fachada creando las interfaces CalibrationService e ImageExtractorService. La motivación principal de esta decisión es disminuir la complejidad del sistema para poder comprenderlo mejor, ya que es muy probable que en un futuro se requiera cambiar la implementación de una de las implementaciones de dichas interfaces para satisfacer nuevas necesidades.

En la figura 5.43 se muestran destacadas las clases que hacen de fachada.

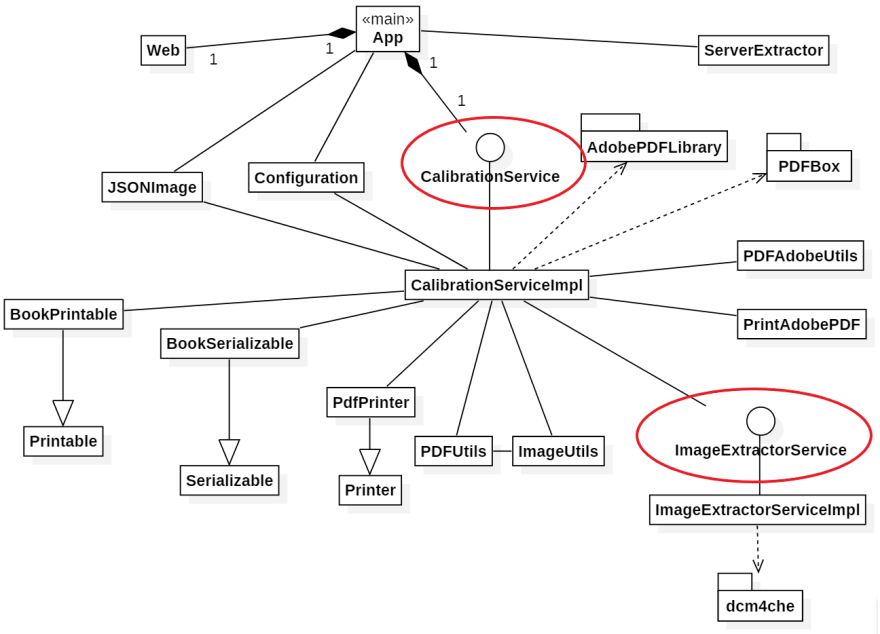


Figura 5.43: Patrones fachada en el sistema

Capítulo 6

Conclusiones

Este trabajo fin de grado se ha realizado sobre un producto que una empresa provee a los hospitales. Dicho producto se trata de un servicio de impresión de imágenes médicas. El objetivo principal del proyecto es investigar la posibilidad de mejorar la impresión de las imágenes médicas, ya que con el sistema actual, muchos estudios al ser impresos pierden detalle debido al cambio de soporte, lo cual hace difícil el diagnóstico. Se debe llegar a un punto donde la imagen impresa se parezca lo máximo posible a la que se muestra en pantalla, para así poder realizar un diagnóstico y una prescripción correctos.

Para alcanzar el objetivo, se ha investigado el proceso de impresión del sistema actual en busca de mejoras para garantizar una mejor impresión. Para realizar dicha investigación, ha sido necesario comprender el estándar DICOM (Digital Imaging and Communication in Medicine), estudiar y comprender las diferentes formas de extraer y manipularlas las imágenes DICOM y buscar la información necesaria para imprimir correctamente dichas imágenes, como pueden ser los formatos de impresión o los espacios, modelos y perfiles de color.

Tras el desarrollo de dicha investigación y la implementación de las mejoras encontradas, se ha concluido que es necesaria una herramienta que permita a un usuario del sistema calibrar una imagen médica antes de imprimirla, para garantizar así que se pueda realizar un correcto diagnóstico en la imagen impresa en papel. Sin embargo, la herramienta de calibración ya existente tiene carencias que la hacen, en ocasiones, difícil de utilizar. Por ello, se ha desarrollado una nueva herramienta con una interfaz web que permite al usuario calibrar una imagen de una forma rápida y sencilla.

Una vez finalizado el proyecto, en algunos estudios aún se puede apreciar alguna tonalidad de color en imágenes que deberían ser impresas en blanco y negro, y en algunas ocasiones la calibración no es suficiente para garantizar un correcto diagnóstico, por lo que el sistema aún es mejorable. Sin embargo, se ha logrado una gran mejora a la hora de calibrar e imprimir las imágenes, permitiendo el diagnóstico y por tanto logrando el objetivo en gran parte de los estudios.

Capítulo 7

Trabajo Futuro

7.1 Auto-Windowing

Muchas veces, con modificar el windowing es suficiente para calibrar una imagen antes de imprimirla, por lo tanto, crear un sistema capaz de indicar unos valores de windowing adecuado ahorraría mucho tiempo al usuario, ya que no tendría que ir analizando los valores de la imagen ni del histograma para hallar los mejores valores de windowing. Se propone diseñar un sistema capaz de indicar al usuario unos valores de windowing automáticos para una imagen dada. Por ejemplo, procesando los valores del pixel data y según la modalidad y la parte del cuerpo examinada, buscar todos los valores de negro que se consideran irrelevantes ya que son ”fondo” y excluirlos del rango del windowing.

7.2 Varias imágenes

Al utilizar la herramienta de calibración, a veces se tiende a calibrar la imagen previsualizada demasiado, lo cual puede producir que el resto de imágenes del estudio dejen de verse correctamente. La herramienta de calibración ha sido creada para en un futuro implementar una solución que permita al usuario ver todas las imágenes de un mismo estudio durante la calibración. Además, yendo un paso más allá, se podría calibrar cada imagen del estudio independientemente de ser necesario, para así evitar que, por ejemplo, la calibración de una imagen afecte al resto.

7.3 Elegir resolución

Debido al escalado que sufre la imagen para que la herramienta de calibración sea fluida, a veces algunos detalles no se pueden apreciar correctamente. Por otro lado, en algunos ordenadores lentos puede que la herramienta no calibre la imagen de previsualización de manera

fluida. Por ello se propone añadir a la herramienta de calibración una opción que permita elegir la resolución de la imagen que se ve como previsualización. Esta solución permitiría ver mejor la imagen a cambio de una mayor lentitud de la herramienta, o usar la herramienta de una forma más fluida a cambio de una peor visualización de la imagen.

7.4 Automatización del sistema de calibración

A pesar de que con esta nueva herramienta de calibración, el proceso es mucho más fluido y sencillo, automatizar dicho proceso ahorraría mucho tiempo a los usuarios, ya que a veces necesitan calibrar varios estudios al día. Por ello se propone crear un sistema inteligente capaz de interpretar la imagen de cada archivo DICOM y generar automáticamente la LUT ideal para dicha imagen. De esta forma el proceso de calibración sería automático.

7.5 Perfil de color propio

Aunque la impresión se ha mejorado y ya no muestra tantas tonalidades de color, se aprecia aún cierta tonalidad en algunos niveles de gris. Con el debido conocimiento sobre los modelos, espacios y perfiles de color y la interpretación del color de cada impresora, podría crearse un perfil de color para cada impresora de tal forma que el book se imprimiese totalmente en blanco y negro a pesar de la utilización de negro compuesto.

Apéndices

Apéndice A

Prototipos

En este capítulo del apéndice se muestran las imágenes del prototipo de la interfaz web usado para extraer los requisitos del sistema.

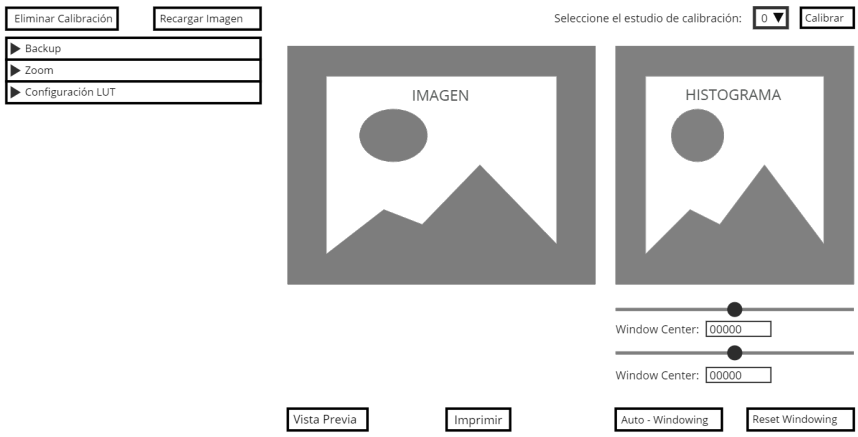


Figura A.1: Página inicial del prototipo

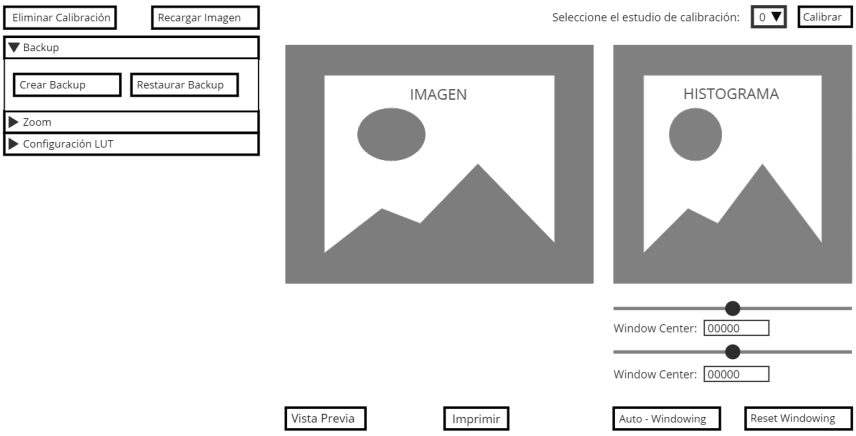


Figura A.2: Página del prototipo con el desplegable Backup

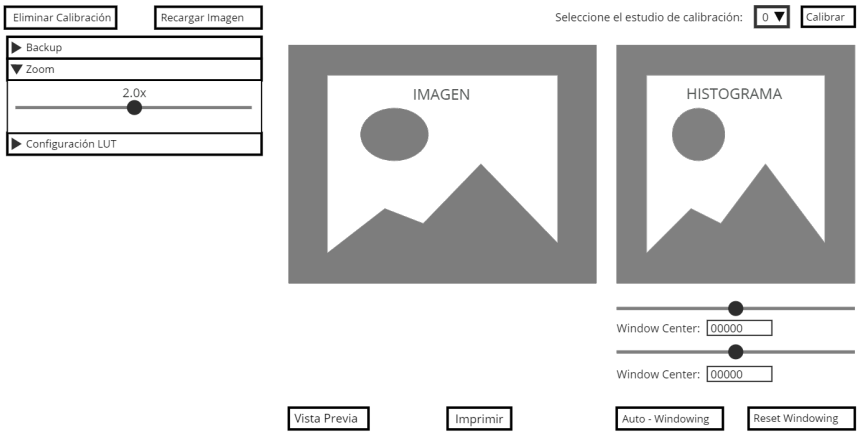


Figura A.3: Página del prototipo con el desplegable Zoom

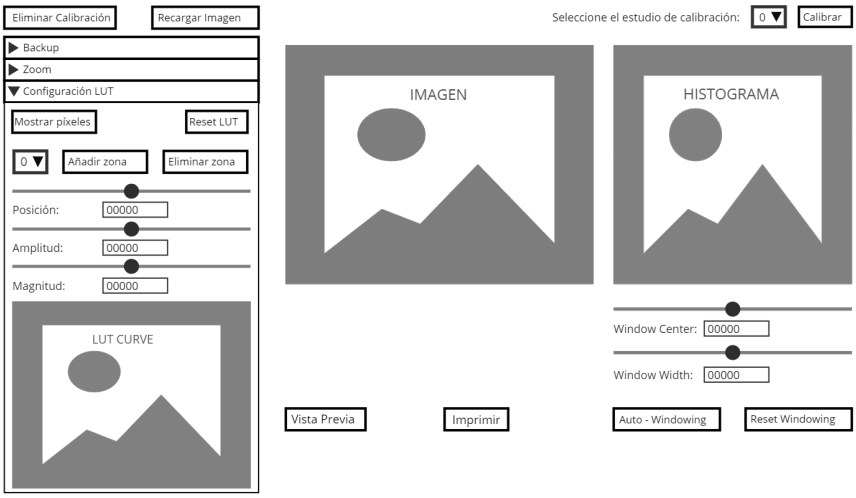


Figura A.4: Página del prototipo con el desplegable LUT

Apéndice B

Casos de Uso

En este capítulo del apéndice se muestran las definiciones de los casos de uso del sistema mediante tablas, como se explica en la subsección 5.2.1 (página 66).

CU-01	
Nombre	Seleccionar estudio
Actor	ACT-01 Usuario
Objetivo	Permitir al ACT-01 Usuario seleccionar el estudio que desea calibrar
Precondiciones	El estudio existe
Pasos	1: El ACT-01 Usuario elige el estudio 2: El sistema pide al ACT-02 Servidor los datos necesarios para la calibración del estudio 3: El ACT-02 Servidor carga el estudio y extrae los datos necesarios para la calibración 4: El sistema recibe los datos y muestra la imagen a calibrar en la pantalla
Excepciones	2.1: Si el ACT-02 Servidor no está disponible, se le muestra un código de error al ACT-01 Usuario 3.1: Si el ACT-02 Servidor no puede extraer los datos necesarios, se le muestra un código de error al ACT-01 Usuario

Tabla B.1: Seleccionar estudio para calibrar

CU-02	
Nombre	Restaurar backup de calibración
Actor	ACT-01 Usuario
Objetivo	Permitir al ACT-01 Usuario cargar los datos de otra calibración previamente guardados. Cabe destacar que la imagen calibrada no forma parte de dichos datos de calibración
Precondiciones	1: Los datos de calibración han sido previamente guardados 2: Hay un estudio cargado
Pasos	1: El ACT-01 Usuario selecciona el backup que desea recuperar 2: El sistema pide al ACT-02 Servidor el backup seleccionado 3: El sistema extrae los datos del backup y los carga sobre el estudio actual
Postcondiciones	Los datos de calibración previos a cargar el backup no se guardan
Excepciones	2.1: Si el ACT-02 Servidor no está disponible, se le muestra un código de error al ACT-01 Usuario 3.1: Si el sistema no puede extraer los datos de calibración, se le muestra un código de error al ACT-01 Usuario

Tabla B.2: Restaurar backup de calibración

CU-03	
Nombre	Crear backup de calibración
Actor	ACT-01 Usuario
Objetivo	Guardar los datos de calibración actuales para continuar la calibración en un futuro o usarlos en estudios similares
Precondiciones	El estudio está cargado
Pasos	1: El sistema crea un backup con los datos de calibración 2: El sistema envía el backup al ACT-02 Servidor para que lo almacene 3: El sistema informa al usuario conforme se ha creado el backup
Postcondiciones	El backup queda almacenado en el ACT-02 Servidor
Excepciones	3.1: Si el backup no ha podido ser creado o el ACT-02 Servidor no está disponible, se le muestra un código de error al ACT-01 Usuario

Tabla B.3: Crear backup de calibración

CU-04	
Nombre	Hacer zoom
Actor	ACT-01 Usuario
Objetivo	Permitir al ACT-01 Usuario ampliar cierta zona de la imagen
Precondiciones	El estudio está cargado
Pasos	1: El ACT-01 Usuario elige el factor de ampliación deseado 2: El ACT-01 Usuario selecciona la zona que desea ampliar 3: El sistema amplia la zona seleccionada

Tabla B.4: Hacer zoom

CU-05	
Nombre	Imprimir Book
Actor	ACT-01 Usuario
Objetivo	Permite al ACT-01 Usuario imprimir un Book
Precondiciones	El estudio está cargado
Pasos	1: El sistema envía al ACT-02 Servidor las opciones de calibración 2: El ACT-02 Servidor calibra el estudio completo y lo imprime
Excepciones	2.1 Si el ACT-02 Servidor no está disponible, se le muestra un código de error al ACT-01 Usuario

Tabla B.5: Imprimir Book

CU-06	
Nombre	Vista previa del Book
Actor	ACT-01 Usuario
Objetivo	Permite al ACT-01 Usuario ver una vista previa del Book correspondiente al estudio cargado actualmente
Precondiciones	El estudio está cargado
Pasos	1: El sistema pide al ACT-02 Servidor el Book y le envía los datos de calibración 2: El ACT-02 Servidor genera un pdf 3: El sistema lee el archivo pdf y se lo muestra al ACT-01 Usuario
Excepciones	1.1 Si el ACT-02 Servidor no está disponible, se le muestra un código de error al ACT-01 Usuario 2.1 Si el ACT-02 Servidor no puede generar el pdf, se le muestra un código de error al ACT-01 Usuario

Tabla B.6: Vista previa del Book

CU-07	
Nombre	Cambiar ventana del histograma
Actor	ACT-01 Usuario
Objetivo	Mostrar al ACT-01 Usuario una serie de herramientas para modificar la ventana del histograma
Precondiciones	El estudio está cargado
Pasos	1: El sistema muestra al ACT-01 Usuario las herramientas disponibles y un gráfico que representa el histograma del estudio actual

Tabla B.7: Cambiar ventana del histograma

CU-08	
Nombre	Cambiar centro de ventana (histograma)
Actor	ACT-01 Usuario
Objetivo	Permitir al ACT-01 Usuario modificar el centro de la ventana del histograma perteneciente al estudio actual
Precondiciones	El estudio está cargado y las herramientas para modificar la ventana del histograma (Tabla B.7) están disponibles
Pasos	1: El ACT-01 Usuario establece el centro de la ventana que desea 2: El sistema modifica el centro de la ventana y actualiza la imagen

Tabla B.8: Cambiar centro de ventana (histograma)

CU-09	
Nombre	Cambiar ancho de ventana (histograma)
Actor	ACT-01 Usuario
Objetivo	Permitir al usuario modificar el ancho de ventana del histograma perteneciente al estudio actual
Precondiciones	El estudio está cargado y las herramientas para modificar la ventana del histograma (Tabla B.7) están disponibles
Pasos	1: El ACT-01 Usuario establece el ancho de ventana que desea 2: El sistema modifica el ancho de ventana y actualiza la imagen

Tabla B.9: Cambiar ancho de ventana (histograma)

CU-10	
Nombre	Calcular auto-windowing
Actor	ACT-01 Usuario
Objetivo	Calcular automáticamente unos parámetros de window width y window center
Precondiciones	El estudio está cargado y las herramientas para modificar la ventana del histograma (Tabla B.7) están disponibles
Pasos	1: Según el estudio cargado, el sistema establece unos valores de window width y wondow center para el estudio actual 2: El sistema actualiza la imagen en pantalla

Tabla B.10: Calcular auto-windowing

CU-11	
Nombre	Reset windowing (histograma)
Actor	ACT-01 Usuario
Objetivo	Reiniciar los valores de window width y window center
Precondiciones	El estudio está cargado y las herramientas para modificar la ventana del histograma (Tabla B.7) están disponibles
Pasos	1: El sistema carga los valores por defecto de window width y window center 2: El sistema actualiza la imagen en pantalla
Postcondiciones	Los valores anteriores de windowing no se guardan

Tabla B.11: Reset windowing (histograma)

CU-12	
Nombre	Configurar LUT
Actor	ACT-01 Usuario
Objetivo	Mostrar al ACT-01 Usuario una serie de herramientas para modificar el LUT
Precondiciones	El estudio está cargado
Pasos	1: El sistema muestra al ACT-01 Usuario las herramientas disponibles y un gráfico que representa la función de LUT

Tabla B.12: Configurar LUT

CU-13	
Nombre	Añadir zona (LUT)
Actor	ACT-01 Usuario
Objetivo	Permite al ACT-01 Usuario crear una ”Zona LUT”
Precondiciones	El estudio está cargado y las herramientas para configurar el LUT están disponibles (Tabla B.12)
Pasos	1a: Si el ACT-01 Usuario selecciona en la imagen los valores de gris que desea modificar, el sistema crea una Zona LUT con unos parámetros que permitan modificar los valores de la zona seleccionada por el ACT-01 Usuario 1b: Si el ACT-01 Usuario no selecciona los valores deseados, se crea una Zona LUT con unos valores por defecto

Tabla B.13: Añadir zona (LUT)

CU-14	
Nombre	Eliminar zona (LUT)
Actor	ACT-01 Usuario
Objetivo	Permite al ACT-01 Usuario eliminar una ”Zona LUT”
Precondiciones	El estudio está cargado, las herramientas para configurar el LUT están disponibles (Tabla B.12) y la zona LUT que se desea eliminar existe
Pasos	1: El ACT-01 Usuario selecciona la zona LUT que desea eliminar 2: El sistema elimina la zona seleccionada 3: El sistema recarga la imagen para que el cambio tome efecto
Postcondiciones	La zona LUT eliminada no se puede recuperar

Tabla B.14: Eliminar zona (LUT)

CU-15	
Nombre	Reiniciar LUT
Actor	ACT-01 Usuario
Objetivo	Poner los valores de LUT por defecto
Precondiciones	El estudio está cargado y las herramientas para configurar el LUT están disponibles (Tabla B.12)
Pasos	1: El sistema elimina todas las Zonas LUT existentes 2: El sistema recarga la imagen para aplicar los cambios
Postcondiciones	Las Zonas LUT eliminadas no se pueden recuperar

Tabla B.15: Reiniciar LUT

CU-16	
Nombre	Mostrar píxeles afectados
Actor	ACT-01 Usuario
Objetivo	Permitir al ACT-01 Usuario ver resaltados los píxeles afectados por una ”Zona LUT”
Precondiciones	El estudio está cargado, las herramientas para configurar el LUT están disponibles (Tabla B.12) y la opción de mostrar los píxeles afectados no está seleccionada
Pasos	1: El ACT-01 Usuario selecciona la Zona LUT que desea ver resaltada 2: El sistema determina los píxeles afectados por dicha zona y modifica la imagen en pantall para resaltar los píxeles correspondientes

Tabla B.16: Mostrar píxeles afectados

CU-17	
Nombre	Ocultar píxeles afectados
Actor	ACT-01 Usuario
Objetivo	Permitir al ACT-01 Usuario dejar de ver resaltados los píxeles afectados por una ”Zona LUT”
Precondiciones	El estudio está cargado, las herramientas para configurar el LUT están disponibles (Tabla B.12) y está seleccionada la opción de ver los píxeles resaltados (Tabla B.16
Pasos	1: El sistema recarga la imagen para dejar de mostrar los píxeles afectados por la Zona LUT actual

Tabla B.17: Ocultar píxeles afectados

CU-18	
Nombre	Cambiar posición (LUT)
Actor	ACT-01 Usuario
Objetivo	Permitir al ACT-01 Usuario modificar la posición de una Zona LUT
Precondiciones	El estudio está cargado y las herramientas para configurar el LUT están disponibles (Tabla B.12) y la Zona LUT que desea modificar existe
Pasos	1: El ACT-01 Usuario selecciona la Zona LUT que desea modificar 2: El ACT-01 Usuario determina el nuevo valor de posición 3: El sistema actualiza la curva LUT acorde con el nuevo valor de posición
Postcondiciones	El sistema modifica todos los píxeles de la imagen afectados

Tabla B.18: Cambiar posición (LUT)

CU-19	
Nombre	Cambiar amplitud (LUT)
Actor	ACT-01 Usuario
Objetivo	Permitir al ACT-01 Usuario modificar la amplitud de una Zona LUT
Precondiciones	El estudio está cargado y las herramientas para configurar el LUT están disponibles (Tabla B.12) y la Zona LUT que desea modificar existe
Pasos	1: El ACT-01 Usuario selecciona la Zona LUT que desea modificar 2: El ACT-01 Usuario determina el nuevo valor de amplitud 3: El sistema actualiza la curva LUT acorde con el nuevo valor de amplitud
Postcondiciones	El sistema modifica todos los píxeles de la imagen afectados

Tabla B.19: Cambiar amplitud (LUT)

CU-20	
Nombre	Cambiar brillo (LUT)
Actor	ACT-01 Usuario
Objetivo	Permitir al ACT-01 Usuario modificar el brillo de una Zona LUT
Precondiciones	El estudio está cargado y las herramientas para configurar el LUT están disponibles (Tabla B.12) y la Zona LUT que desea modificar existe
Pasos	1: El ACT-01 Usuario selecciona la Zona LUT que desea modificar 2: El ACT-01 Usuario determina el nuevo valor de brillo 3: El sistema actualiza la curva LUT acorde con el nuevo valor de brillo
Postcondiciones	El sistema modifica todos los píxeles de la imagen afectados

Tabla B.20: Cambiar brillo (LUT)

CU-21	
Nombre	Cambiar contraste (LUT)
Actor	ACT-01 Usuario
Objetivo	Permitir al ACT-01 Usuario modificar el contraste de una Zona LUT
Precondiciones	El estudio está cargado y las herramientas para configurar el LUT están disponibles (Tabla B.12) y la Zona LUT que desea modificar existe
Pasos	1: El ACT-01 Usuario selecciona la Zona LUT que desea modificar 2: El ACT-01 Usuario determina el nuevo valor de contraste 3: El sistema actualiza la curva LUT acorde con el nuevo valor de contraste
Postcondiciones	El sistema modifica todos los píxeles de la imagen afectados

Tabla B.21: Cambiar contraste (LUT)

Apéndice C

Clases

En este capítulo del anexo se reflejan las imágenes que muestran las clases del sistema con sus atributos y métodos. Imágenes C.1 y C.2.

Figura C.1: Clases pertenecientes al sub-sistema, con atributos y métodos

Figura C.2: Clases pertenecientes al sistema (sin sub-sistema), con atributos y métodos

Apéndice D

Diagramas de Flujo de Datos

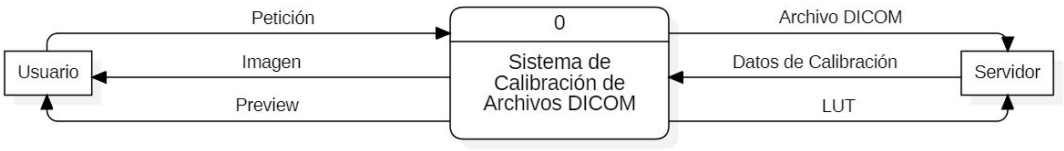


Figura D.1: Nivel 0

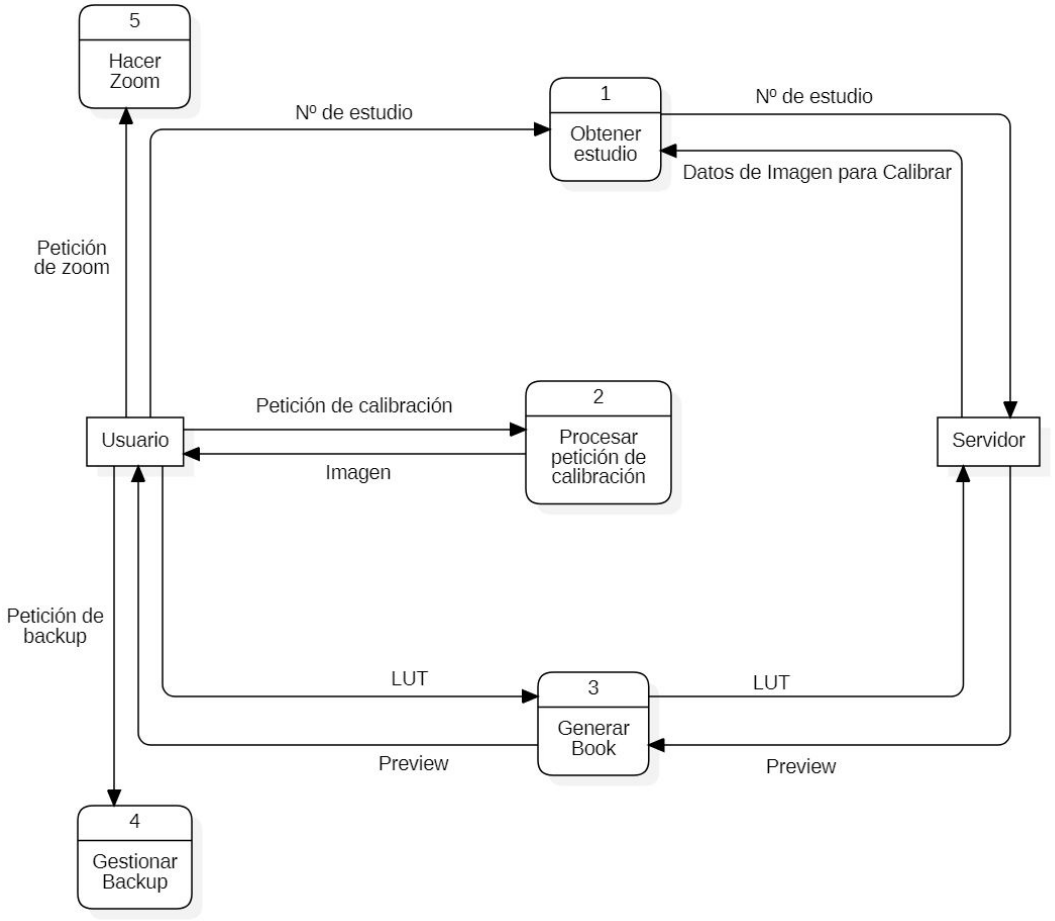


Figura D.2: Nivel 1

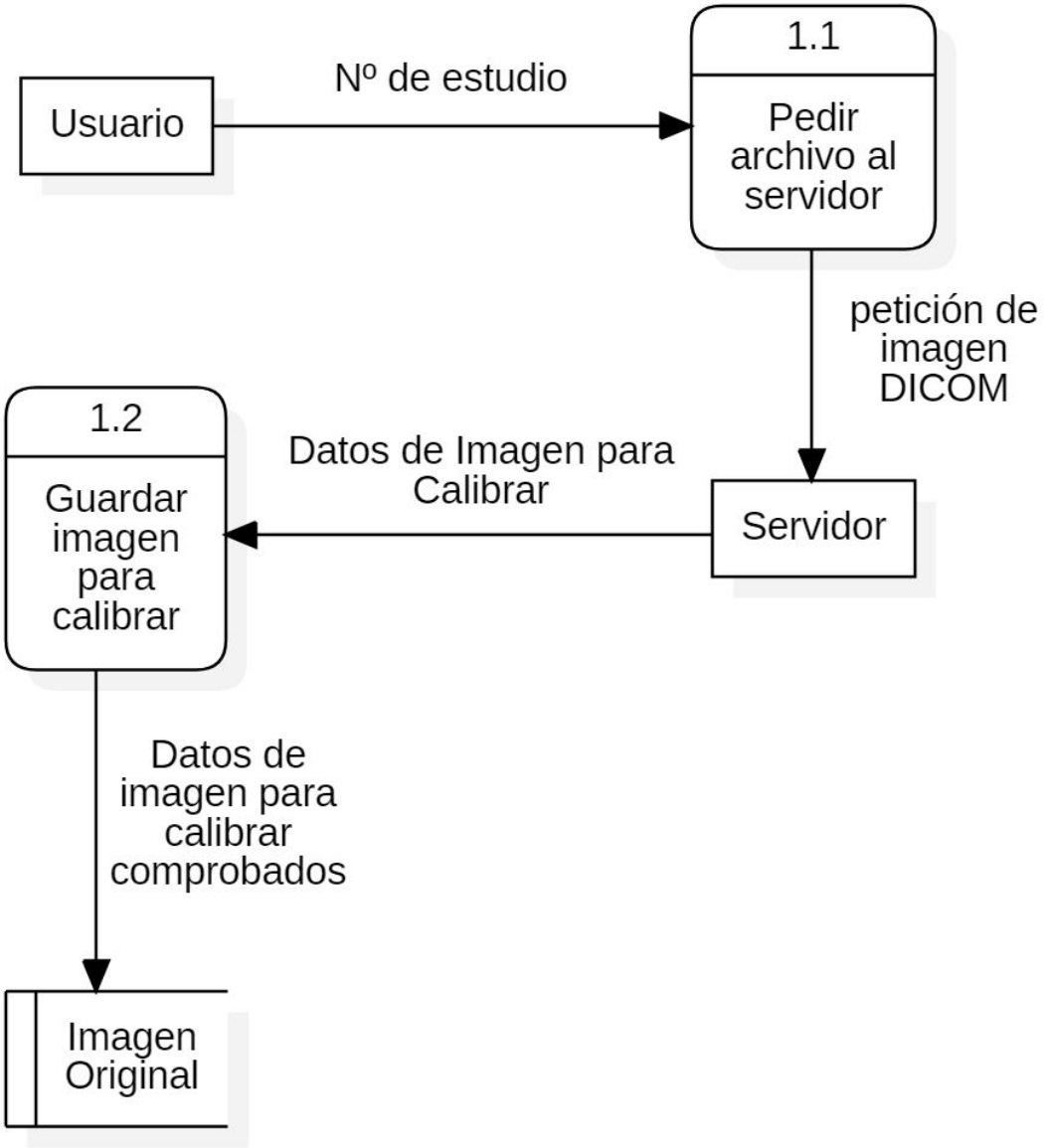


Figura D.3: Nivel 2

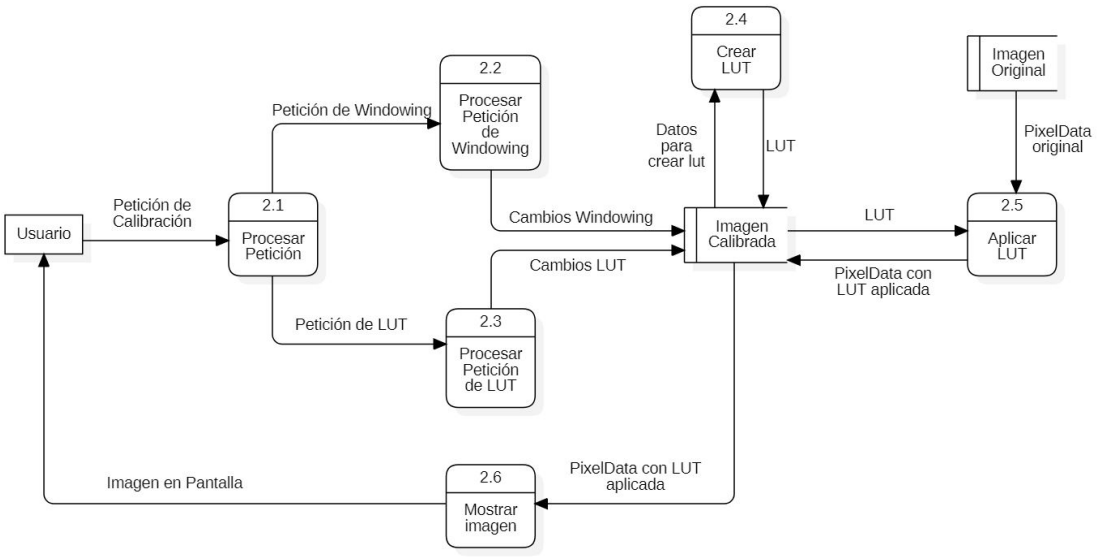


Figura D.4: Nivel 2

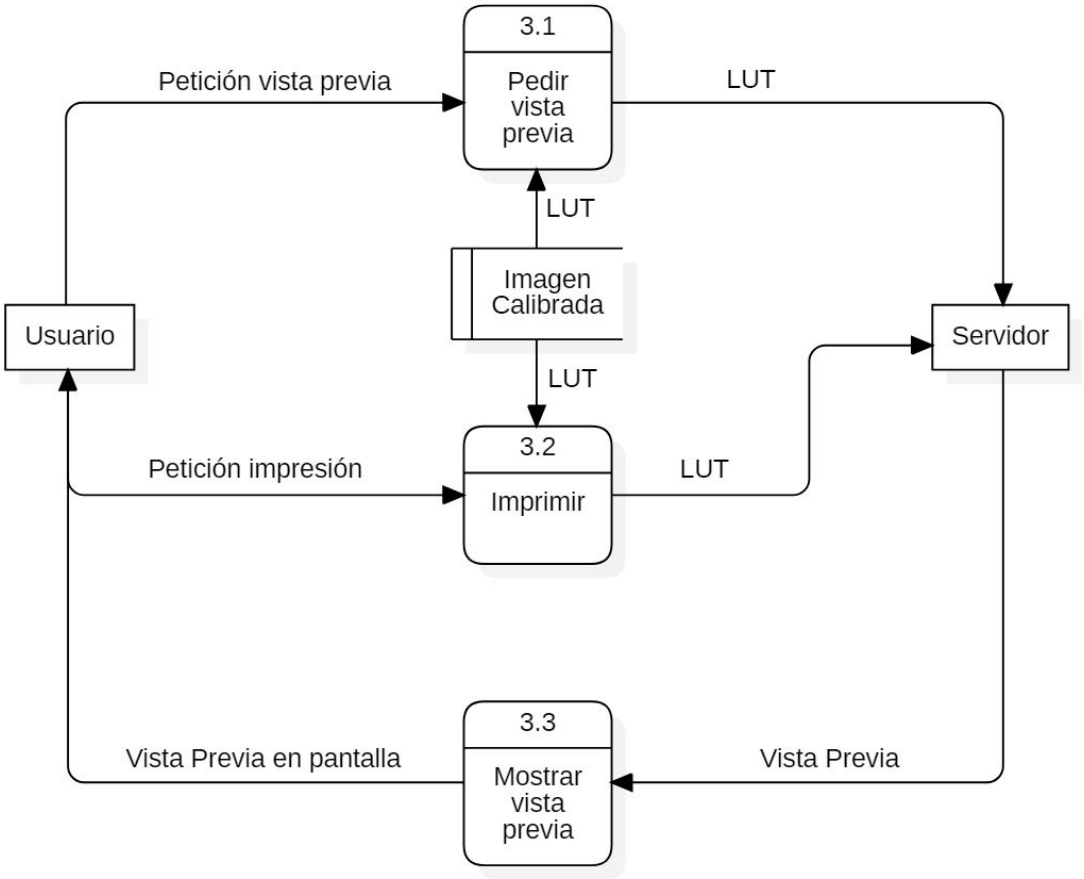


Figura D.5: Nivel 2

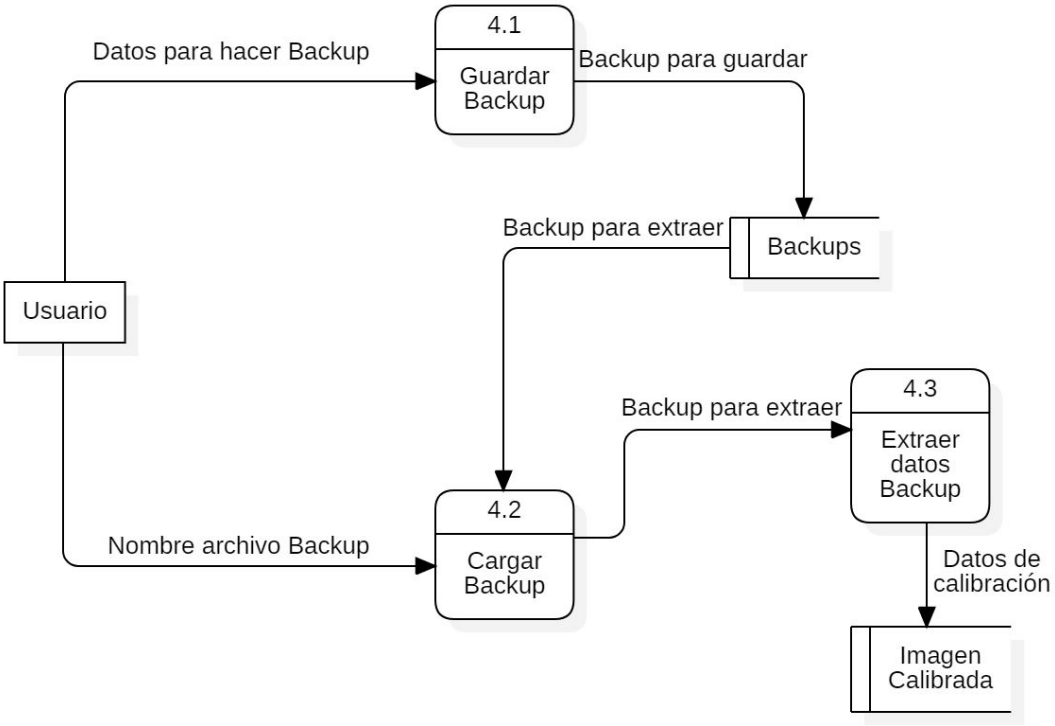


Figura D.6: Nivel 2

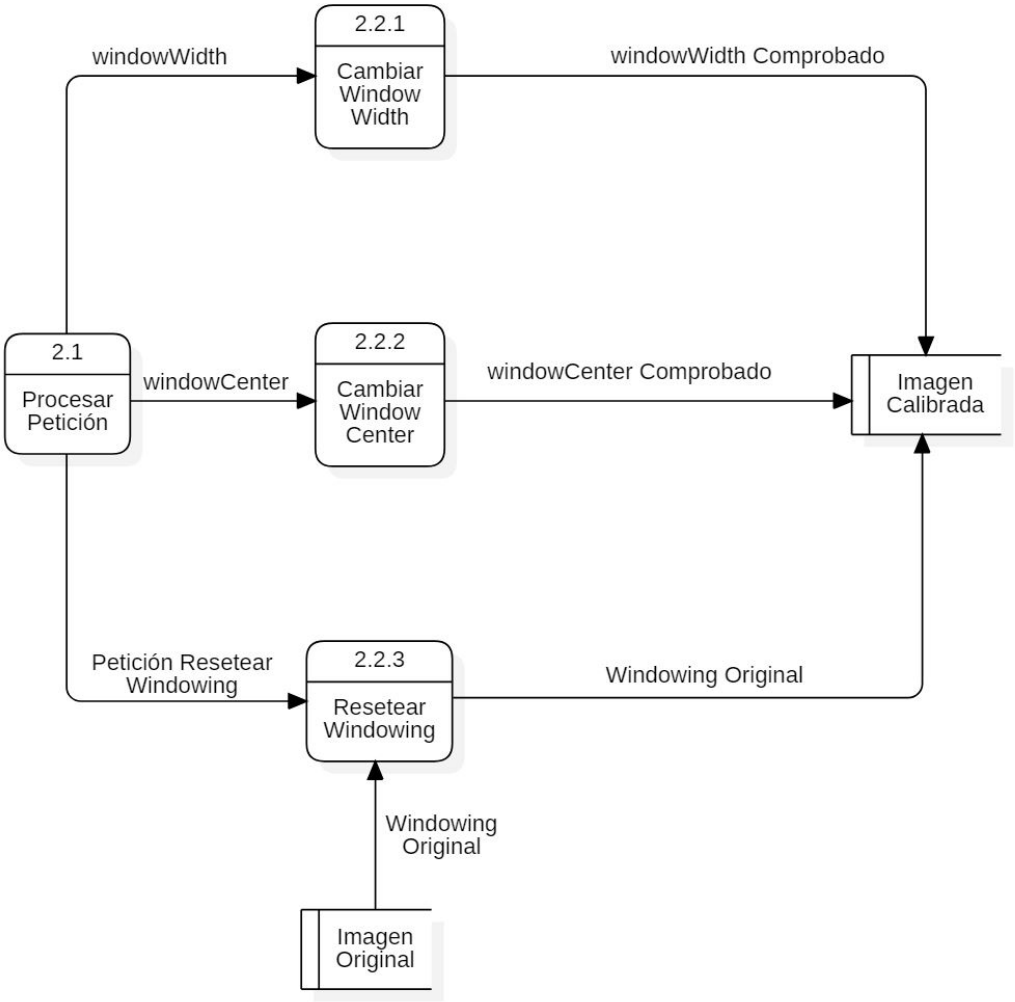


Figura D.7: Nivel 3

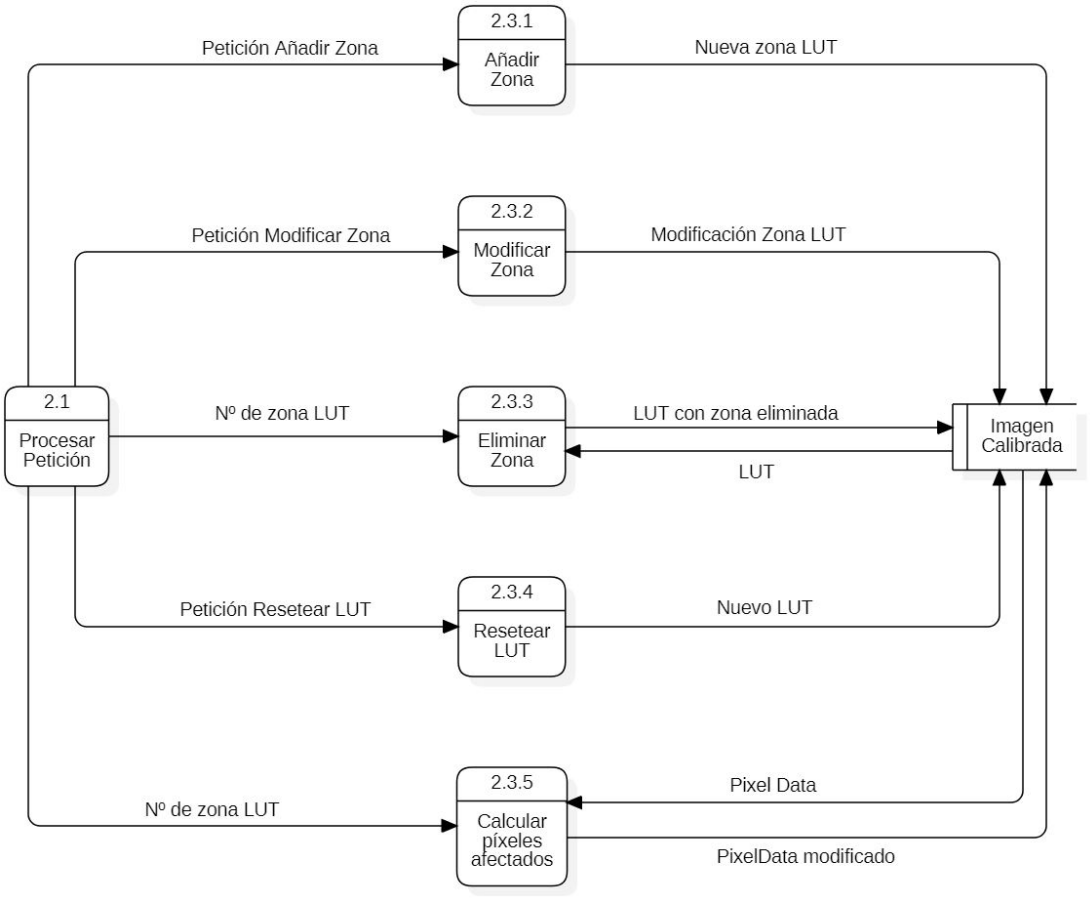


Figura D.8: Nivel 3

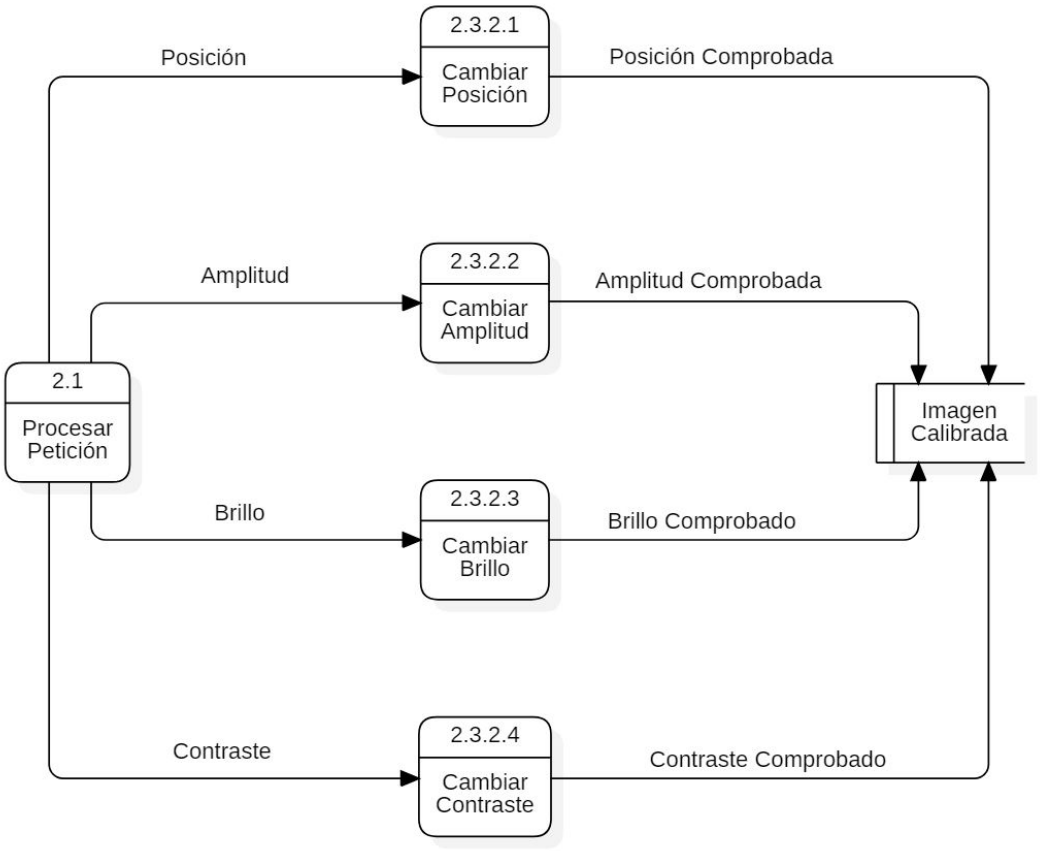


Figura D.9: Nivel 4

Apéndice E

Especificaión de procesos

Proceso 1.1: Pedir archivo al servidor	
Entrada	Nº de estudio.
Salida	Petición de datos de imagen.
Descripción	El proceso recibe el número de estudio a calibrar y envía al servidor una petición.

Tabla E.1: Proceso 1.1

Proceso 1.2: Guardar imagen para calibrar	
Entrada	Datos de imagen para calibrar
Salida	Datos de imagen para calibrar comprobados
Descripción	El proceso recibe los datos de la imagen que se desea calibrar, los procesa y los guarda en el almacén

Tabla E.2: Proceso 1.2

Proceso 2.1: Procesar Petición	
Entrada	Petición de calibración
Salida	Petición de Windowing
Salida	Petición de LUT
Descripción	El proceso recibe la petición de calibración, comprueba que los datos recibidos son correctos y se los envía al proceso correspondiente.

Tabla E.3: Proceso 2.1

Proceso 2.2.1: Cambiar Window Width	
Entrada	Window Width
Salida	Window Width comprobado
Descripción	Comprueba que e valor de entrada es correcto y lo guarda en el almacén

Tabla E.4: Proceso 2.2.1

Proceso 2.2.2: Cambiar Window Center	
Entrada	Window Center
Salida	Window Center comprobado
Descripción	Comprueba que e valor de entrada es correcto y lo guarda en el almacén

Tabla E.5: Proceso 2.2.2

Proceso 2.2.3: Resetear Windowing	
Entrada	Petición resetear windowing Windowing original
Salida	Windowing original
Descripción	Cuando recibe la petición de resetear el windowing, coge del almacén Imagen Original el windowing original y lo guarda en el almacén Imagen Calibrada.

Tabla E.6: Proceso 2.2.3

Proceso 2.3.1: Añadir Zona	
Entrada	Petición añadir zona
Salida	Nueva zona LUT
Descripción	Cuando recibe la petición, crea una nueva zona LUT con valores por defecto y la guarda en el almacén

Tabla E.7: Proceso 2.3.1

Proceso 2.3.2.1: Cambiar posición	
Entrada	Posición
Salida	Posición comprobada
Descripción	Comprueba que el valor de entrada es correcto y lo guarda en el almacén

Tabla E.8: Proceso 2.3.2.1

Proceso 2.3.2.2: Cambiar amplitud	
Entrada	Amplitud
Salida	Amplitud comprobada
Descripción	Comprueba que e valor de entrada es correcto y lo guarda en el almacén

Tabla E.9: Proceso 2.3.2.2

Proceso 2.3.2.3: Cambiar brillo	
Entrada	Brillo
Salida	Brillo comprobado
Descripción	Comprueba que el valor de entrada es correcto y lo guarda en el almacén

Tabla E.10: Proceso 2.3.2.3

Proceso 2.3.2.4: Cambiar contraste	
Entrada	Contraste
Salida	Contraste comprobado
Descripción	Comprueba que el valor de entrada es correcto y lo guarda en el almacén

Tabla E.11: Proceso 2.3.2.4

Proceso 2.3.3: Eliminar zona	
Entrada	Número de zona LUT LUT
Salida	LUT con zona eliminada
Descripción	Recibe el número de zona que se debe eliminar, recoge el LUT del almacén, elimina la zona LUT y guarda el LUT de nuevo en el almacén.

Tabla E.12: Proceso 2.3.3

Proceso 2.3.4: Resetear LUT	
Entrada	Petición resetear LUT
Salida	Nuevo LUT
Descripción	Cuando recibe la petición, crea una nueva LUT y la guarda en el almacén.

Tabla E.13: Proceso 2.3.4

Proceso 2.3.5: Calcular píxeles afectados	
Entrada	número de zona LUT Pixel Data
Salida	Pixel Data modificado
Descripción	Coge el Pixel Data del almacén, comprueba el número de la zona LUT recibida y transforma el Pixel Data de forma que se muestren los píxeles afectados por dicha Zona LUT. Guarda el Pixel Data modificado en el almacén.

Tabla E.14: Proceso 2.3.5

Proceso 2.4: Crear LUT	
Entrada	Datos para crear LUT
Salida	LUT
Descripción	El proceso recoge del almacén los datos necesarios para crear la LUT, la crea y la almacena.

Tabla E.15: Proceso 2.4

Proceso 2.5: Aplicar LUT	
Entrada	LUT Pixel Data original
Salida	Pixel Data con LUT aplicada
Descripción	Recoge de los almacenes la LUT y el Pixel Data de la imagen original y aplica la LUT sobre el Pixel Data, para luego guardarlo en el almacén.

Tabla E.16: Proceso 2.5

Proceso 2.6: Mostrar Imagen	
Entrada	Pixel Data con LUT aplicada
Salida	Imagen en pantalla
Descripción	Muestra por pantalla el Pixel Data interpolado al usuario como una imagen en escala de grises.

Tabla E.17: Proceso 2.6

Proceso 3.1: Pedir vista previa	
Entrada	Petición vista previa LUT
Salida	LUT
Descripción	Cuando recibe la petición de vista previa, recoge la LUT del almacén y se la envía al servidor.

Tabla E.18: Proceso 3.1

Proceso 3.2: Imprimir	
Entrada	Petición de impresión LUT
Salida	LUT
Descripción	Cuando recibe la petición de impresión, recoge la LUT del almacén y se la envía al servidor.

Tabla E.19: Proceso 3.2

Proceso 3.3: Mostrar vista previa	
Entrada	Vista previa
Salida	Vista previa en pantalla
Descripción	Recibe el Book del servidor y lo muestra por pantalla.

Tabla E.20: Proceso 3.3

Proceso 4.1: Guardar Backup	
Entrada	Datos para hacer backup
Salida	Backup para guardar
Descripción	Genera un archivo de backup con los datos de entrada y lo almacena en el almacén

Tabla E.21: Proceso 4.1

Proceso 4.2: Cargar Backup	
Entrada	Nombre archivo backup Backup para extraer
Salida	Backup para extraer
Descripción	Busca en el almacén el backup correspondiente al nombre de archivo que recibe como entrada y lo manda como salida

Tabla E.22: Proceso 4.2

Proceso 4.3: Extraer Datos Backup	
Entrada	Backup para extraer
Salida	Datos de calibración
Descripción	Extrae los datos de calibración del backup y los guarda en el almacén.

Tabla E.23: Proceso 4.3

Apéndice F

Diagramas de Secuencia

En este anexo se muestran todos los diagramas de secuencia de todo el sistema, como se indica en las secciones 5.1.2 y 5.2.2 (páginas 61 y 71).

F.1 Sistema de calibración

El diagrama de la imagen F.2 es aplicable para las interacciones CambiarWindowWidth, CambiarWindowCenter, CambiarPosicion, CambiarAmplitud, CambiarBrillo y CambiarContraste.

Las imágenes pertenecientes a esta sección son: F.1, F.2, F.3, F.4, F.5, F.6, F.7, F.8, F.9, F.10, F.11, F.12, F.13, F.14, F.15 y F.16.

F.2 Sistema de impresión

Las imágenes pertenecientes a esta sección son: F.17, F.18, F.19, F.20, F.21, F.22 y F.23.

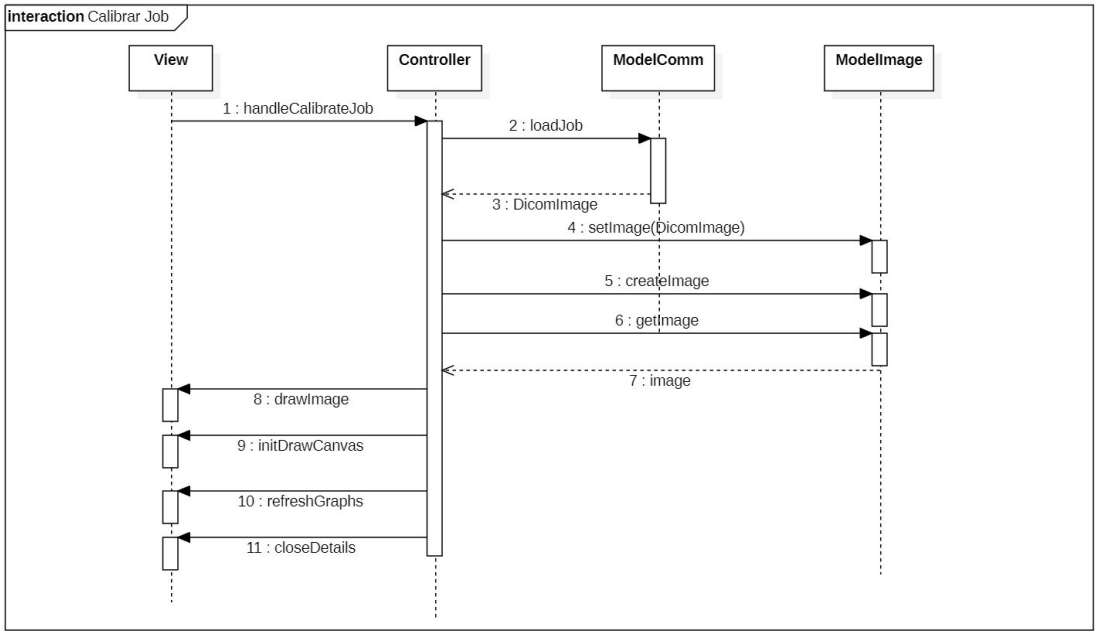


Figura F.1: Interacción Calibrar Job

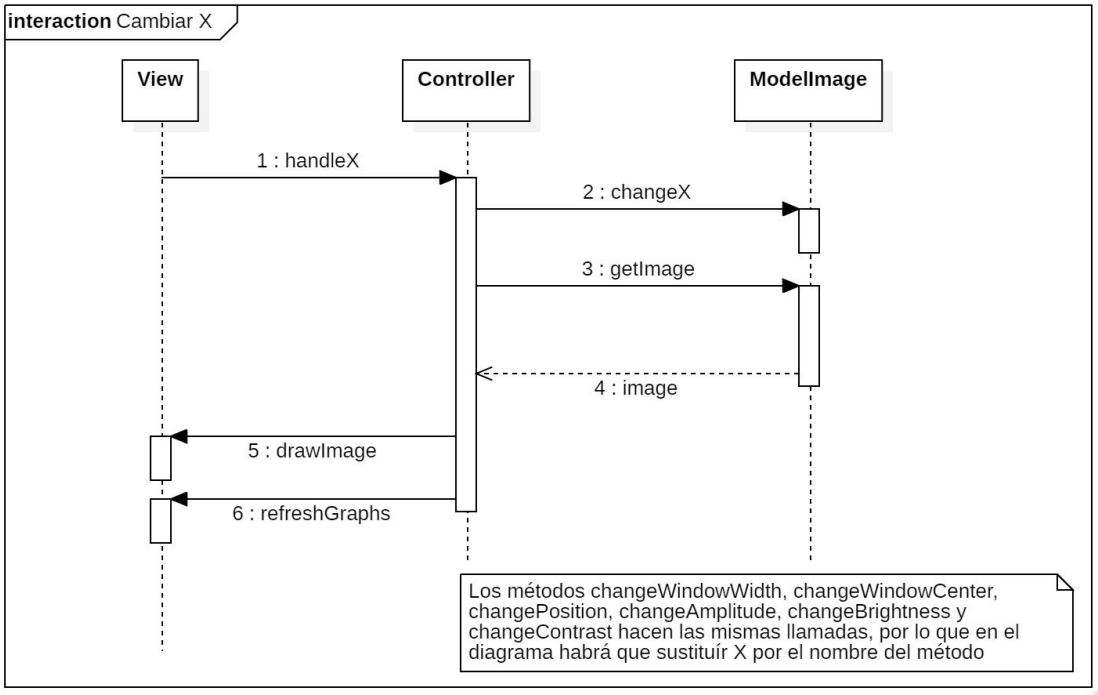


Figura F.2: Interacción Cambiar X

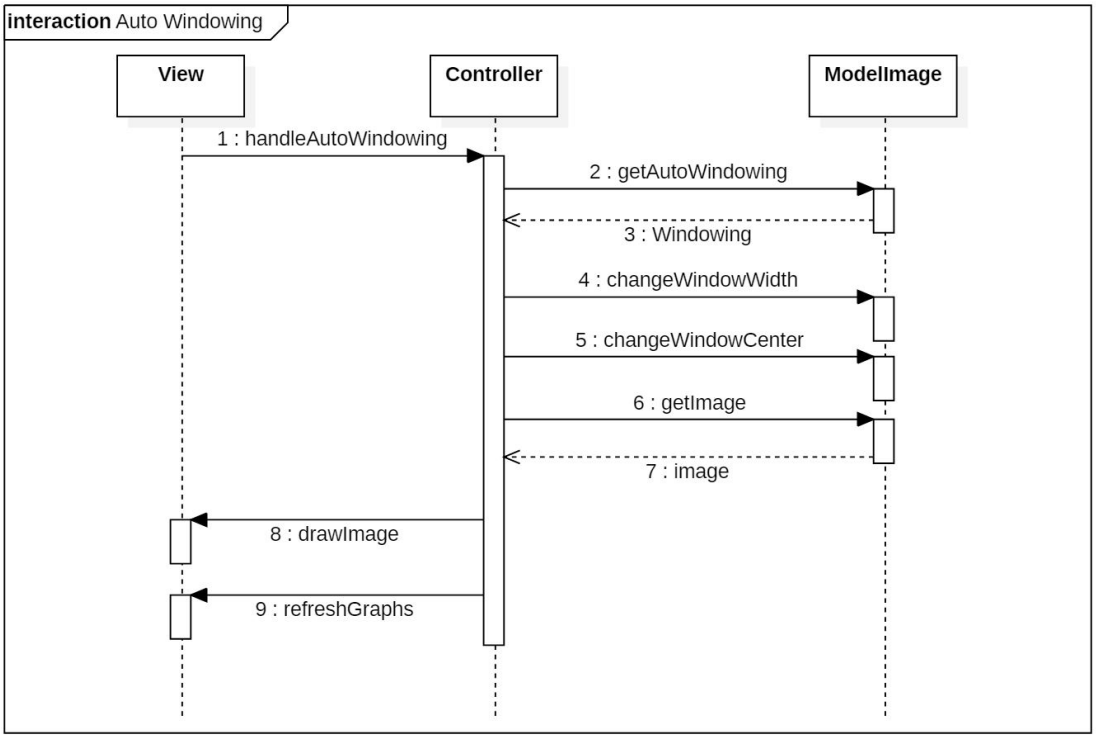


Figura F.3: Interacción Auto-Windowing

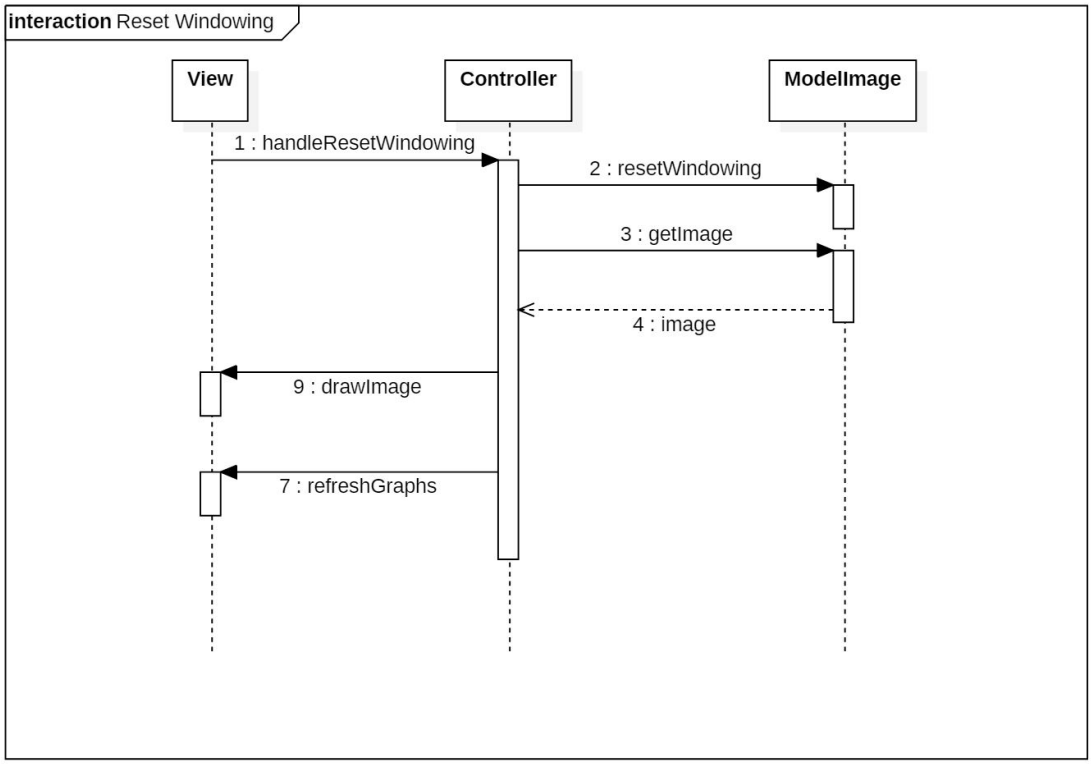


Figura F.4: Interacción Reiniciar Windowing

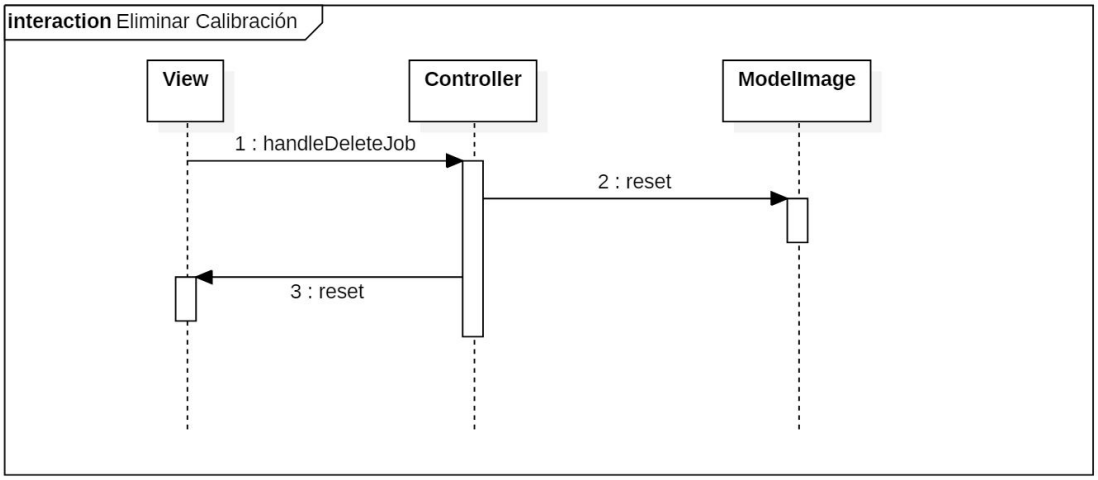


Figura F.5: Interacción Eliminar Calibración

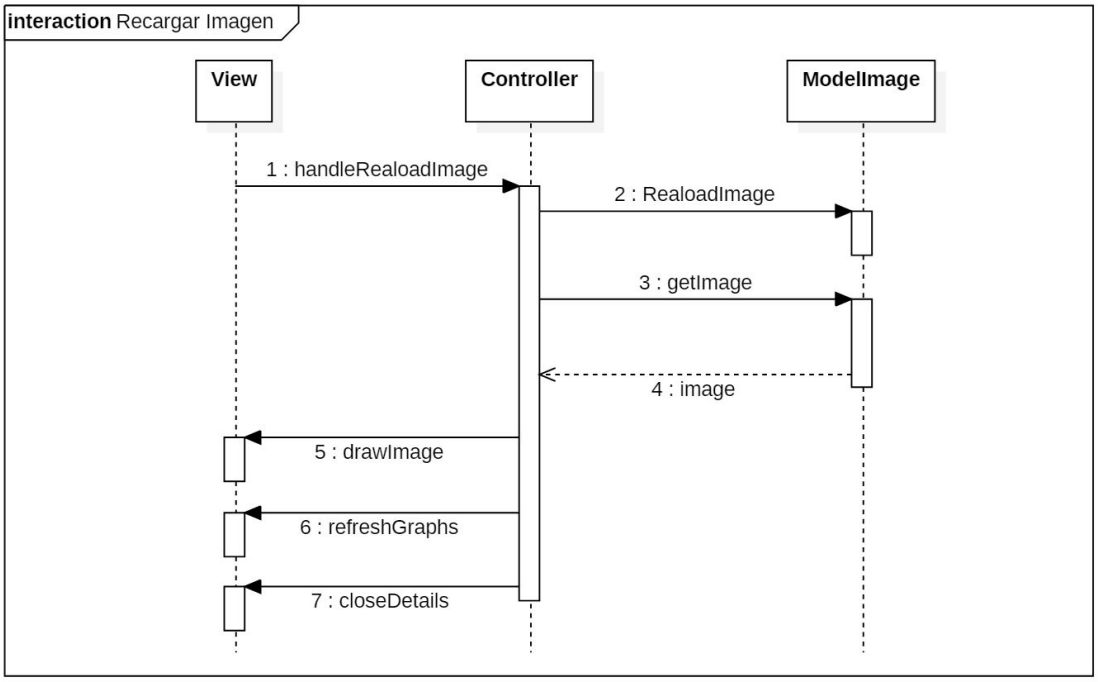


Figura F.6: Interacción Recargar Imagen

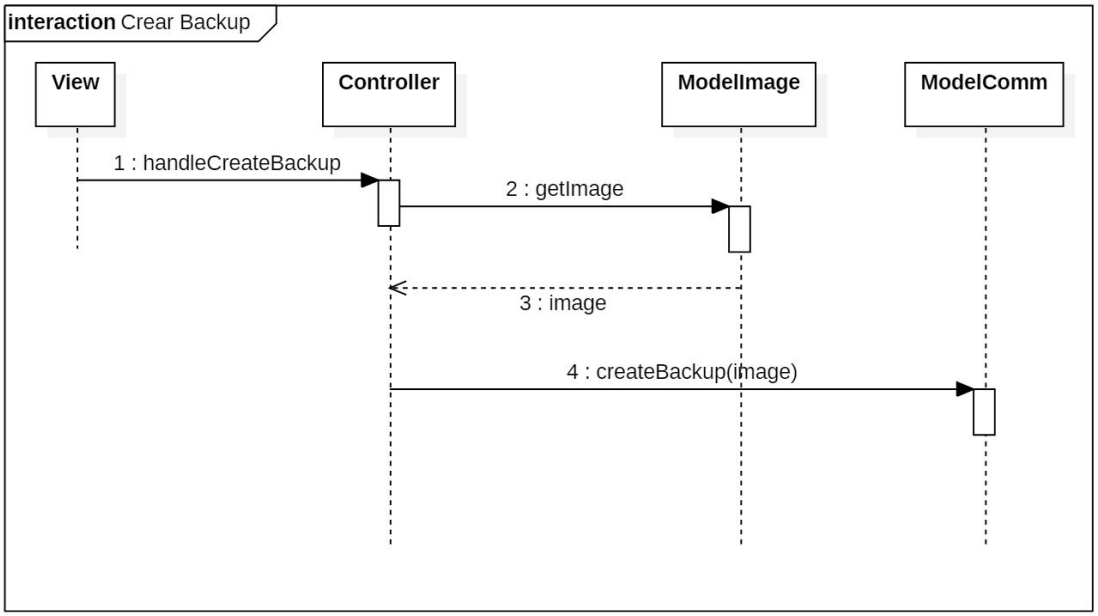


Figura F.7: Interacción Crear Backup

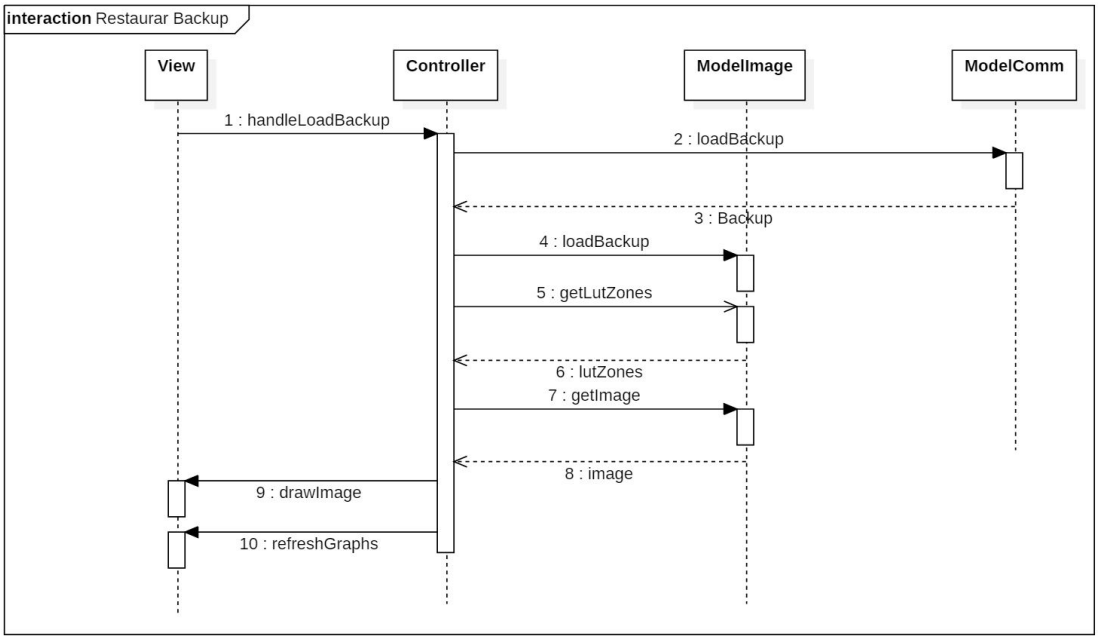


Figura F.8: Interacción Restaurar Backup

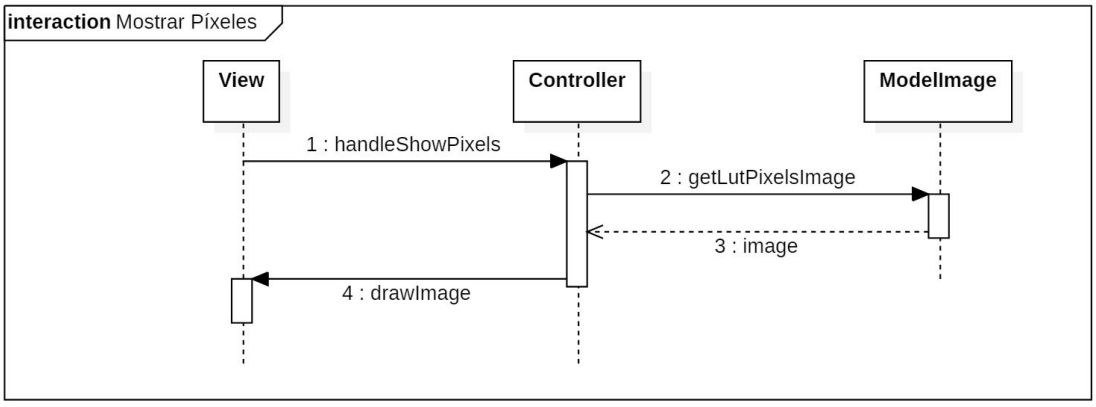


Figura F.9: Interacción Mostrar Píxeles

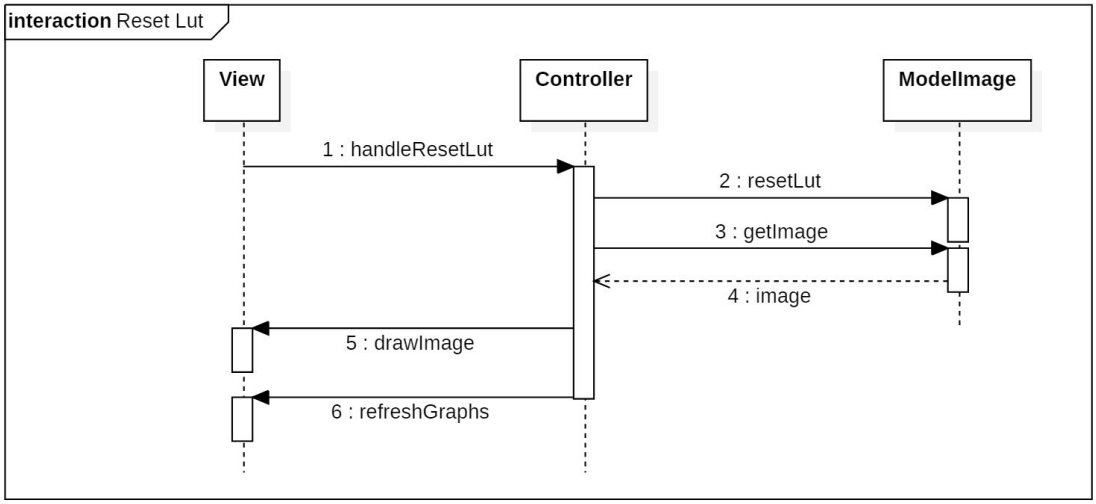


Figura F.10: Interacción Reiniciar LUT

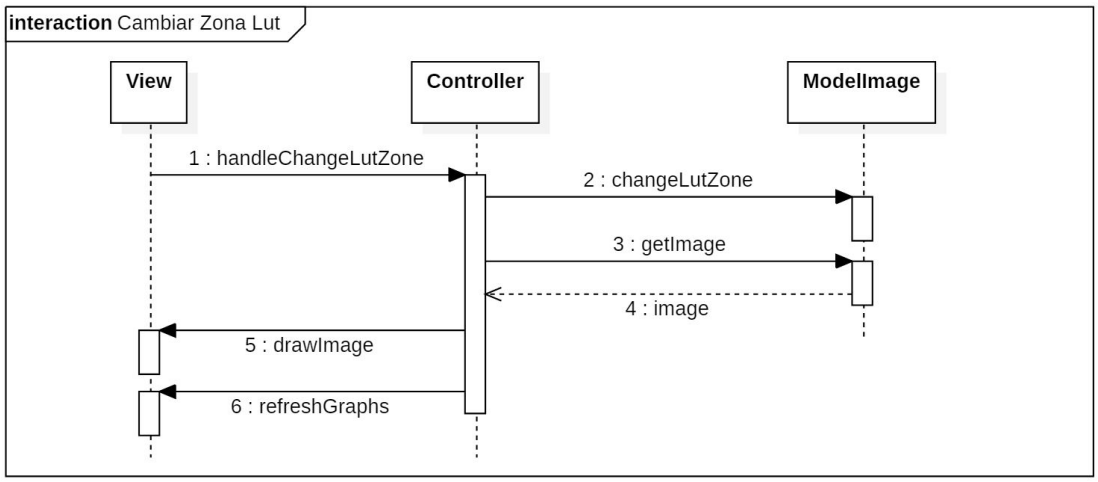


Figura F.11: Interacción Cambiar Zona LUT

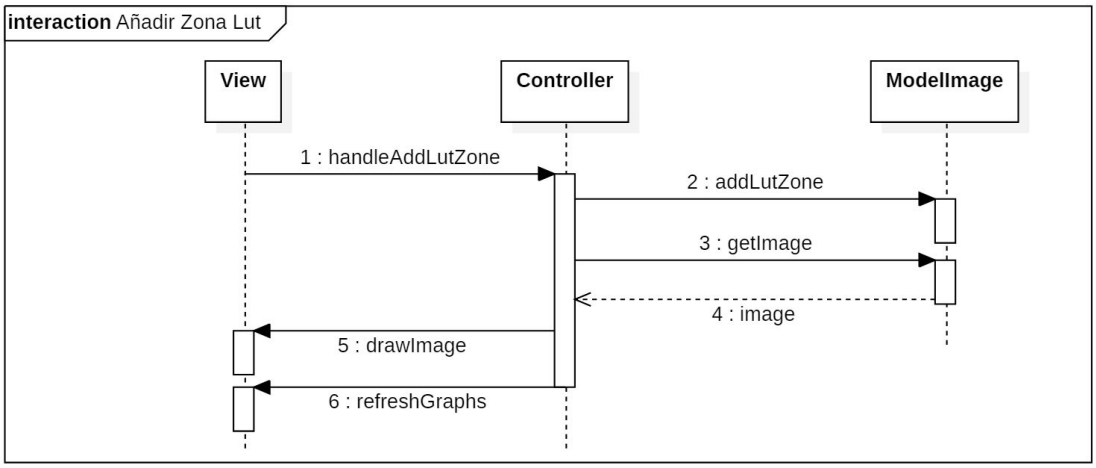


Figura F.12: Interacción Añadir Zona LUT

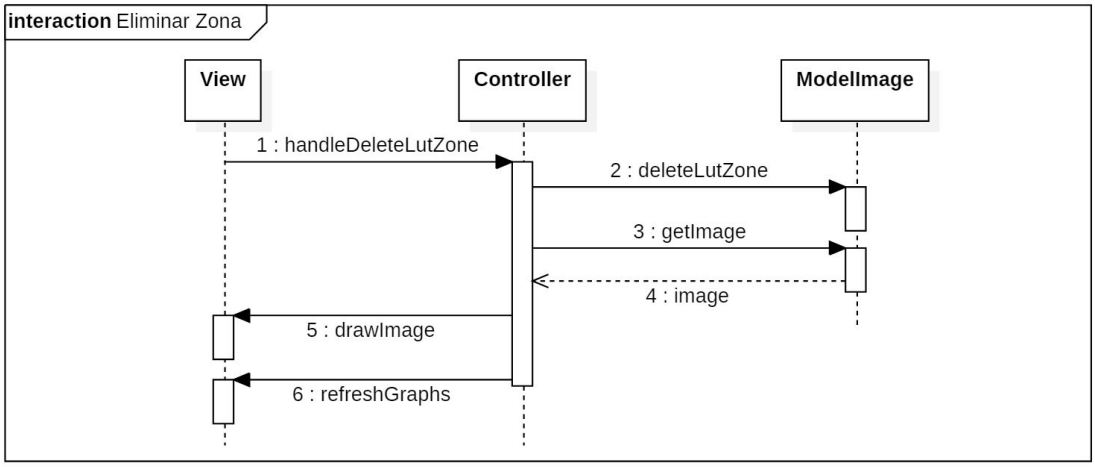


Figura F.13: Interacción Eliminar Zona LUT

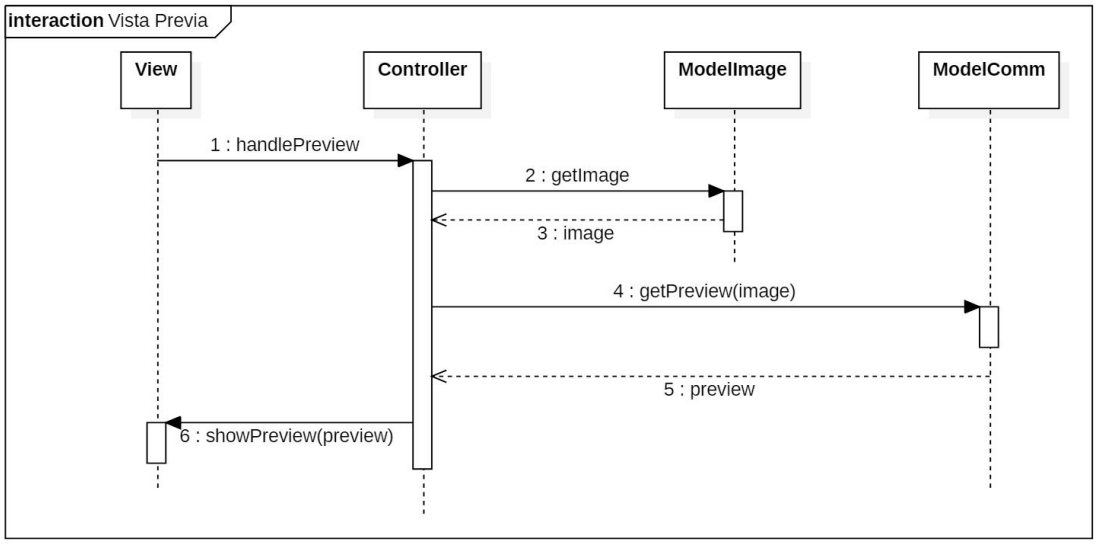


Figura F.14: Interacción Vista Previa

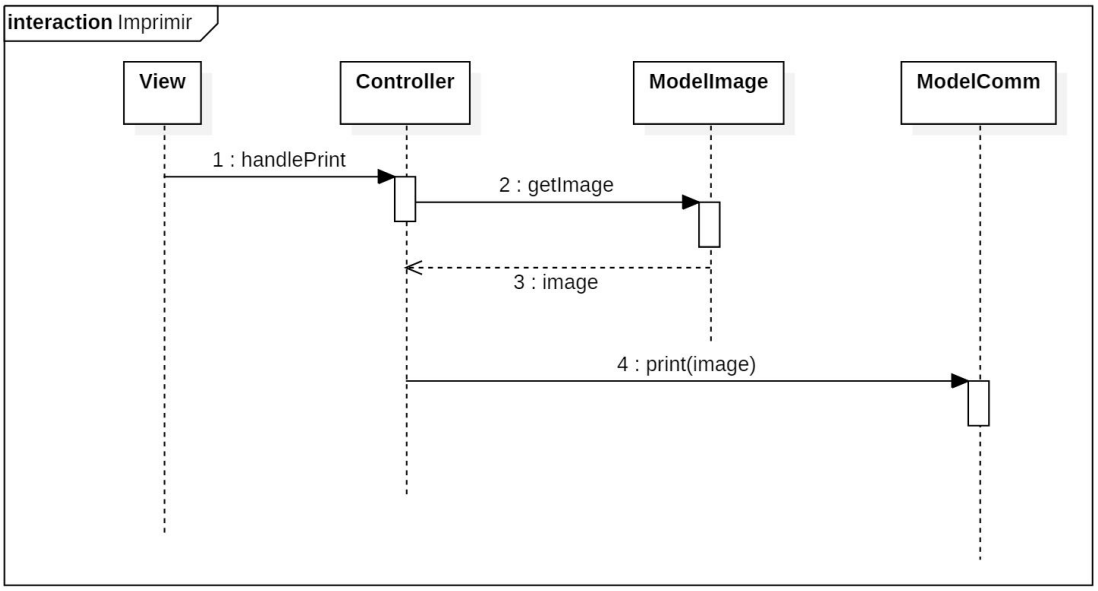


Figura F.15: Interacción Imprimir

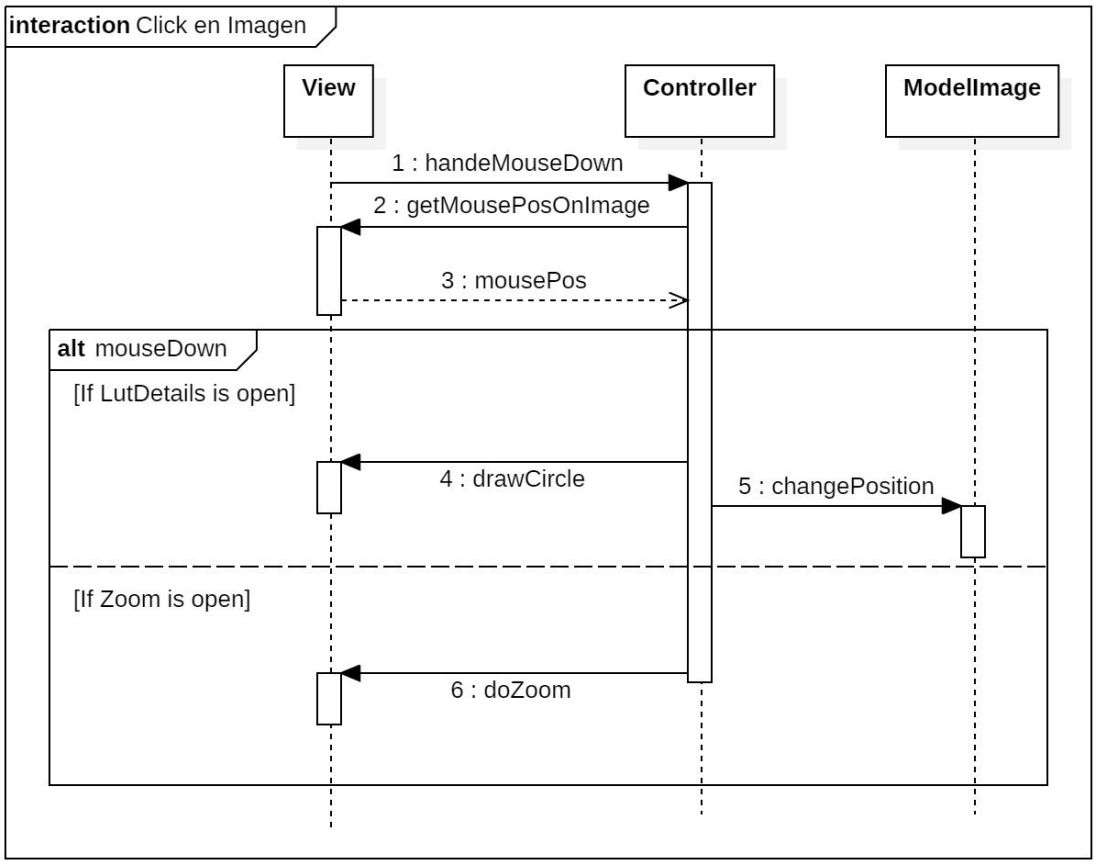


Figura F.16: Interacción Click en la Imagen

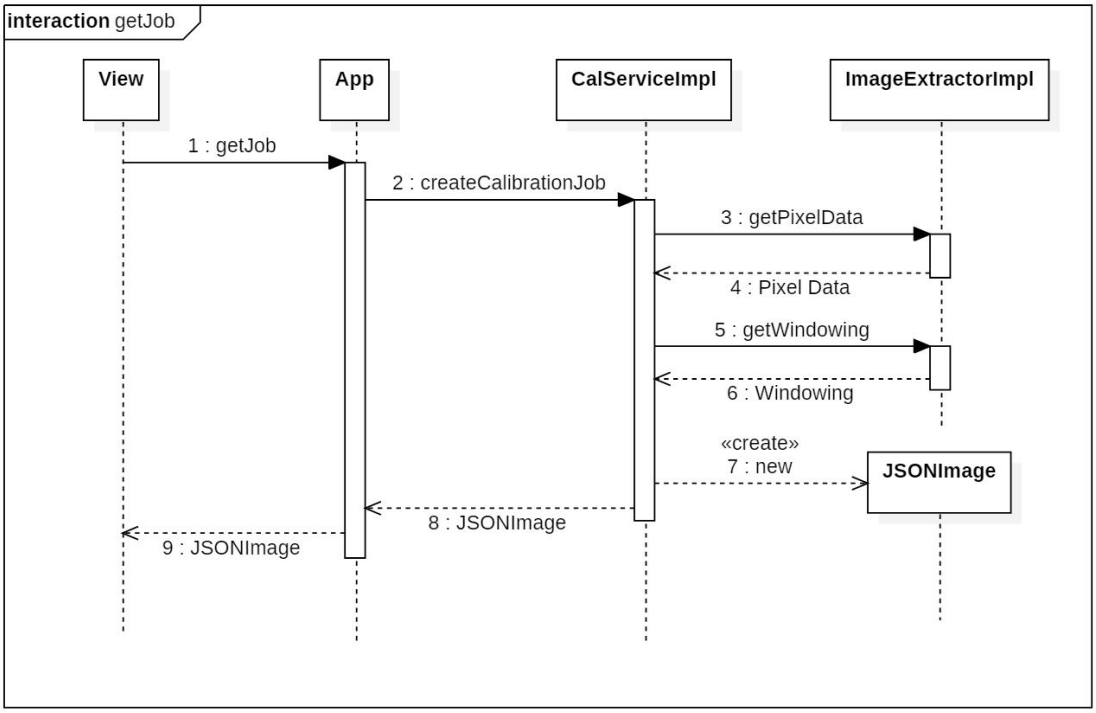


Figura F.17: Interacción Get Job

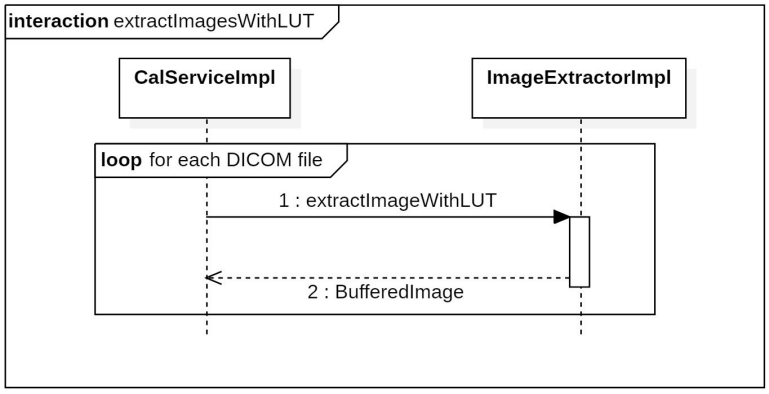


Figura F.18: Interacción Extraer Imagen con LUT aplicado

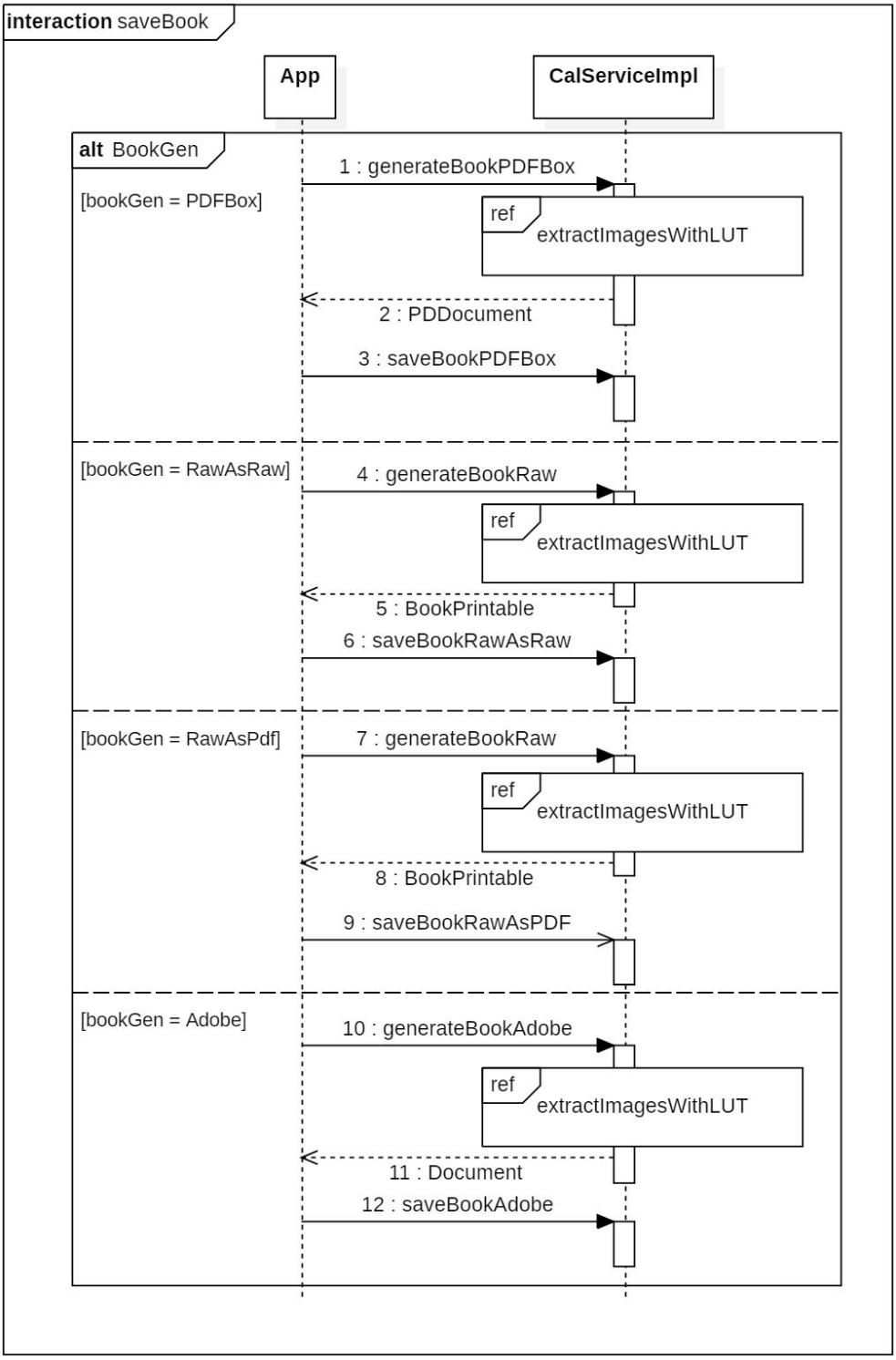


Figura F.19: Interacción Guardar Book

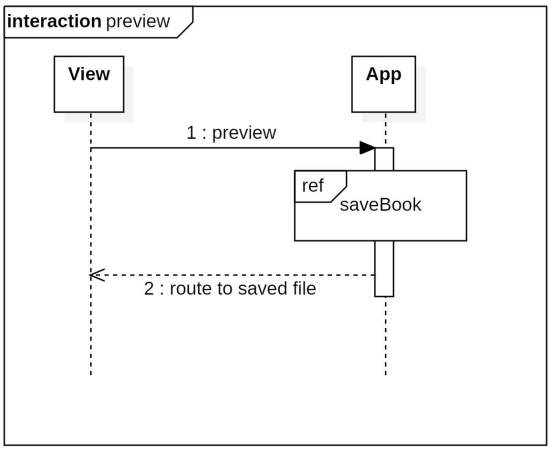


Figura F.20: Interacción Vista Previa

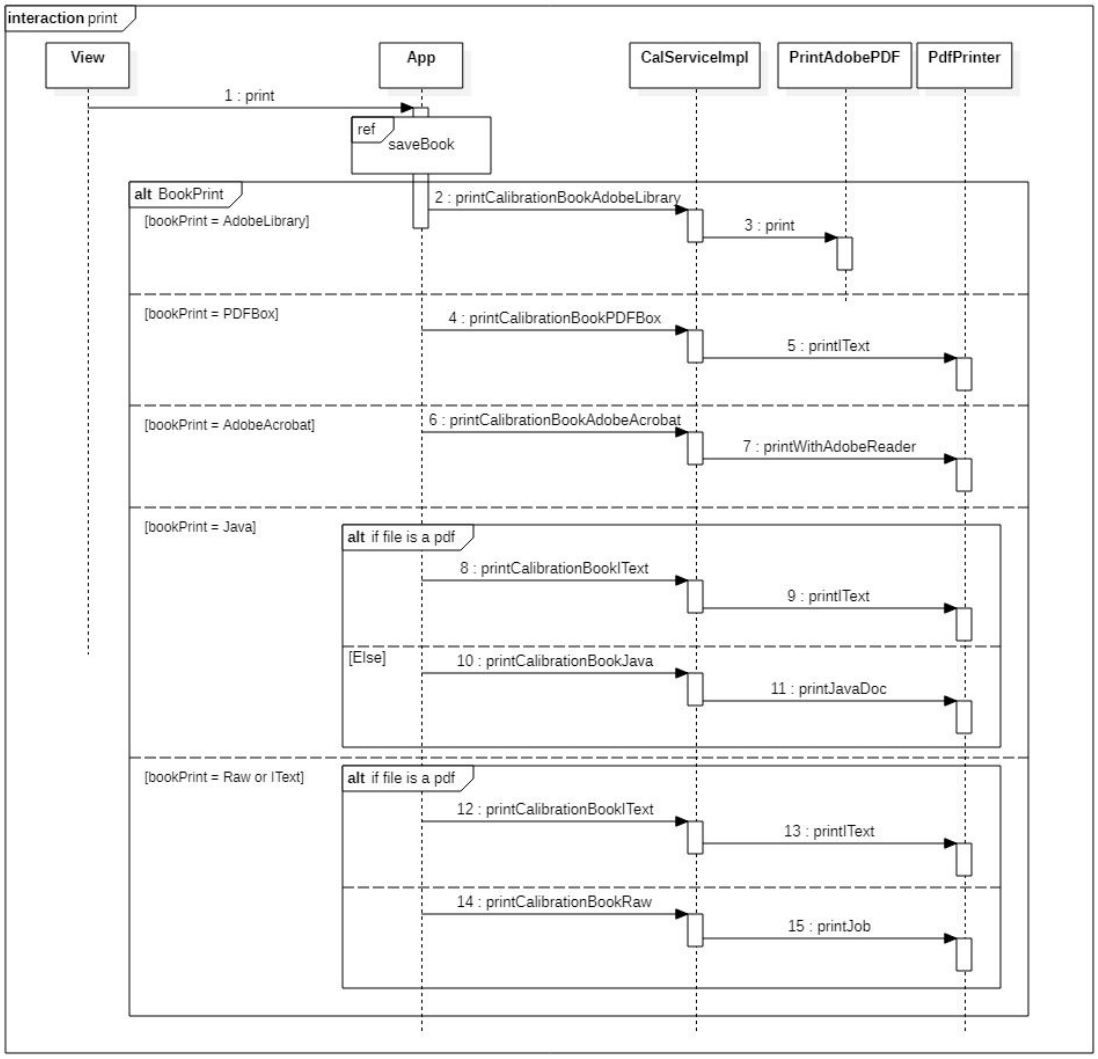


Figura F.21: Interacción Imprimir

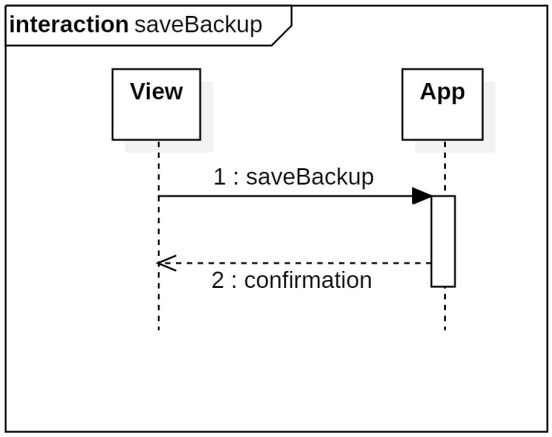


Figura F.22: Interacción Guardar Backup

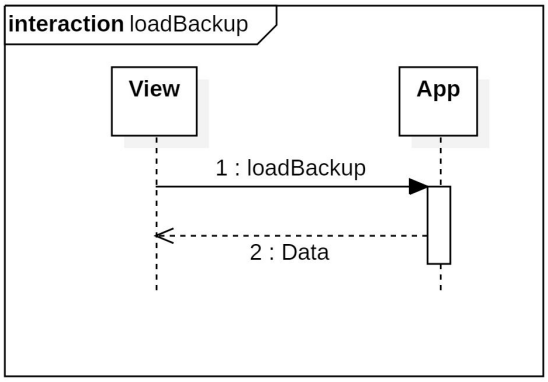


Figura F.23: Interacción Cargar Backup

Lista de acrónimos

LUT *Look Up Table.*

UCR *Under Color Removal.*

ICC *International Color Consortium.*

RGB *Red Green Blue.*

CMYK *Cian Magenta Yellow Key.*

MVC *Model View Controller.*

Bibliografía

[1] "Dicom standard." [En línea]. Disponible en: https://www.dicomstandard.org/

[2] "Dicom tags standard." [En línea]. Disponible en: http://dicom.nema.org/medical/dicom/current/output/html/part06.html

[3] "Dicom modalities." [En línea]. Disponible en: https://www.dicomlibrary.com/dicom/modality/

[4] "Java.com." [En línea]. Disponible en: https://www.java.com/

[5] "Oracle java qué es." [En línea]. Disponible en: https://www.oracle.com/technetwork/topics/newtojava/downloads/index.html

[6] "Java wikipedia." [En línea]. Disponible en: https://en.wikipedia.org/wiki/Java_(programming_language)

[7] "History and license - python.org." [En línea]. Disponible en: https://docs.python.org/3/license.html

[8] "Python wikipedia." [En línea]. Disponible en: https://en.wikipedia.org/wiki/Python_(programming_language)

[9] "Javascript | mdn." [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[10] "Ecmascript wikipedia." [En línea]. Disponible en: https://en.wikipedia.org/wiki/ECMAScript

[11] "Html wikipedia." [En línea]. Disponible en: https://en.wikipedia.org/wiki/HTML

[12] "The html and css standards world wide web consortium." [En línea]. Disponible en: https://www.w3.org/standards/webdesign/htmlcss

[13] "World wide web consortium (w3c)." [En línea]. Disponible en: https://www.w3.org/

[14] "Css wikipedia." [En línea]. Disponible en: https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[15] "Spring web." [En línea]. Disponible en: https://spring.io/

[16] "Spring wikipedia." [En línea]. Disponible en: https://en.wikipedia.org/wiki/Spring_Framework

[17] "Spring aop." [En línea]. Disponible en: https://docs.spring.io/spring/docs/2.5.x/reference/aop.html

[18] "Jest - delightful javascript testing." [En línea]. Disponible en: https://jestjs.io/

[19] "Junit 5." [En línea]. Disponible en: https://junit.org/junit5/

[20] "Apache tomcat." [En línea]. Disponible en: http://tomcat.apache.org/

[21] "Apache software foundation." [En línea]. Disponible en: https://www.apache.org/

[22] "The apache http server project." [En línea]. Disponible en: https://httpd.apache.org/

[23] "Open source clinical image and object management." [En línea]. Disponible en: https://www.dcm4che.org/

[24] "D3.js - data-driven documents." [En línea]. Disponible en: https://d3js.org/

[25] "Jquery." [En línea]. Disponible en: https://jquery.com/

[26] "Jquery w3schools." [En línea]. Disponible en: https://www.w3schools.com/jquery/

[27] "Spyder website." [En línea]. Disponible en: https://www.spyder-ide.org/

[28] "Spyder (software) - wikipedia." [En línea]. Disponible en: https://en.wikipedia.org/wiki/Spyder_(software)

[29] "Visual studio code." [En línea]. Disponible en: https://code.visualstudio.com/

[30] "Visual studio code 1.0." [En línea]. Disponible en: https://code.visualstudio.com/blogs/2016/04/14/vscode-1.0

[31] "Intelligent code completion." [En línea]. Disponible en: https://en.wikipedia.org/wiki/Intelligent_code_completion

[32] "Staruml." [En línea]. Disponible en: http://staruml.io/

[33] "Gimp - gnu image manipulation." [En línea]. Disponible en: https://www.gimp.org/

[34] "Vectr - free online vector graphics editor." [En línea]. Disponible en: https://vectr.com/

[35] "Latex - a document preparation system." [En línea]. Disponible en: https://www.latex-project.org/

[36] "Overleaf, online latex editor." [En línea]. Disponible en: https://www.overleaf.com/

[37] "Home | libreoffice - free office suite." [En línea]. Disponible en: https://www.libreoffice.org/

[38] "Weasis medical viewer." [En línea]. Disponible en: https://nroduit.github.io/en/

[39] "Conquest dicom software." [En línea]. Disponible en: https://ingenium.home.xs4all.nl/dicom.html

[40] "Xerox color 550/560/570." [En línea]. Disponible en: https://www.xerox.es/es-es/impresion-digital/prensas-digitales/xerox-color-550-560

[41] "Xerox altalink c8030/c8035/c8045/c8055/c8070." [En línea]. Disponible en: https://www.xerox.es/es-es/oficina/impresoras-multifuncion/altalink-c8000-series/specifications

[42] "Xerox phaser 7760." [En línea]. Disponible en: https://www.xerox.es/es-es/oficina/impresoras/phaser-7760

[43] "Workplace an digital printing solutions." [En línea]. Disponible en: https://www.xerox.com/

[44] "Uml use case diagrams overview." [En línea]. Disponible en: https://www.uml-diagrams.org/use-case-diagrams.html

[45] A. Cockburn, *Writing Effective Use Cases.* Addison-Wesley, 2001.

[46] "Uml class and object diagrams overview." [En línea]. Disponible en: https://www.uml-diagrams.org/class-diagrams-overview.html

[47] "Uml sequence diagrams overview." [En línea]. Disponible en: https://www.uml-diagrams.org/sequence-diagrams.html

[48] "Observer (patrón de diseño) - wikipedia." [En línea]. Disponible en: https://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o)

[49] "Current edition - dicom standard." [En línea]. Disponible en: https://www.dicomstandard.org/current/

[50] "International color consortium." [En línea]. Disponible en: http://color.org/

[51] A. Syromiatnikov and D. Weyns, "A journey through the land of model-view-design patterns," 2014. [En línea]. Disponible en: https://www.semanticscholar.org/paper/A-Journey-through-the-Land-of-Model-View-Design-Syromiatnikov-Weyns/360c2dc46bb1814d118a4dcf73bf6690b78f3201

[52] "Model-view-controller," 2014. [En línea]. Disponible en: http://wiki.c2.com/?ModelViewController